Position Paper: From Edge AI to Adaptive Edge AI

Fabrizio Pittorino and Manuel Roveri
Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, Italy
Email: {fabrizio.pittorino, manuel.roveri}@polimi.it

***Abstract*—Edge AI is often framed as model compression and deployment under tight constraints. We argue a stronger operational thesis: *Edge AI in realistic deployments is necessarily adaptive*. In long-horizon operation, a fixed (non-adaptive) configuration faces a fundamental failure mode: as data and operating conditions evolve and change in time, it must either (i) violate time-varying budgets (latency/energy/thermal/connectivity/privacy) or (ii) lose predictive reliability (accuracy and, critically, calibration), with risk concentrating in transient regimes and rare time intervals rather than in average performance. If a deployed system cannot *reconfigure* its computation - and, when required, its model state - under evolving conditions and constraints, it reduces to static embedded inference and cannot provide sustained utility. This position paper introduces a minimal Agent-System-Environment (ASE) lens that makes adaptivity precise at the edge by specifying (i) what changes, (ii) what is observed, (iii) what can be reconfigured, and (iv) which constraints must remain satisfied over time. Building on this framing, we formulate ten research challenges for the next decade, spanning theoretical guarantees for evolving systems, dynamic architectures and hybrid transitions between data-driven and model-based components, fault/anomaly-driven targeted updates, System-1/System-2 decompositions (anytime intelligence), modularity, validation under scarce labels, and evaluation protocols that quantify lifecycle efficiency and recovery/stability under drift and interventions.**

***Index Terms*—Edge AI, adaptivity, dynamic neural networks, continual learning, anytime inference, resource constraints**

I. Edge AI is already adaptive (operationally)

Edge AI is often presented as *deploying smaller models under tight constraints* - compression, quantization, and hardware-aware inference. A complementary (and increasingly dominant) view treats the edge as a *runtime regime*: on-device models must operate over time under distribution shift, heterogeneous hardware/software stacks, and time-varying budgets, and they increasingly exploit conditional computation (routing, early exits, selective activation) as a primary efficiency mechanism [1], [2]. In realistic deployments, the operating conditions evolve continuously: data distributions drift (context shifts, sensor aging), resource budgets fluctuate (battery, thermals, latency), connectivity is intermittent, and user requirements change. In short, *edge systems operate under non-stationarity and time-varying constraints.*

This has immediate consequences for non-adaptive systems. A static deployment fixes a single operating point - a model configuration, an execution policy, an implicit calibration state - and then assumes that the feasibility region remains stable. When conditions drift, that assumption breaks in measurable ways. First, a fixed configuration can become *infeasible*: thermal throttling, workload interference, or reduced battery headroom can push latency/energy beyond budgets even if the model is unchanged. Second, even when execution remains feasible, predictive *reliability* degrades under shift: accuracy drops, and uncertainty may become miscalibrated, so the system may remain overconfident while wrong. Third, long-horizon operation concentrates risk in *transients* and *rare time intervals* rather than in average metrics: brief regimes (e.g., low light, motion blur, sensor drift, connectivity loss) dominate operational failures unless explicitly handled. These failure modes motivate the central claim of this paper: sustained Edge AI cannot be reduced to static embedded inference.

We therefore advance the following position statement:

> *Edge AI and Adaptive Edge AI should be considered semantically equivalent as operational concepts.*

If *edge* implies sustained operation in the wild, then adaptivity - the ability to reconfigure computation and, when required, update model state as conditions evolve under constraints - is not an optional add-on but a defining requirement.

This equivalence forces explicit assumptions about (i) *what changes* (data, task, hardware, user), (ii) *what is observable*, (iii) *what can be reconfigured*, and (iv) *how systems should be evaluated* over long horizons with drift and interventions. These questions are increasingly central in the literature: surveys on test-time adaptation highlight the breadth of deployment-time shifts [3], on-device benchmarks emphasize that adaptation must be assessed under realistic device constraints and operating conditions [4], and energy/latency measurement itself requires standardized reporting to be comparable across platforms [5].

Our goal is to align the community on a coherent agenda. We contribute:

1) **Thesis:** Static deployment is operationally incomplete for Edge AI; adaptivity is intrinsic.
2) **Framework:** A minimal agent-system-environment lens that clarifies what changes, what is observed, what can be reconfigured, and what must remain satisfied over time.
3) **Agenda:** Ten research challenges for the next decade, with evaluation protocols and measurable success criteria.

II. From Edge AI to Adaptive Edge AI

A useful way to interpret current practice is in terms of *when* optimization happens. Most Edge AI pipelines optimize *before* deployment: models are compressed/quantized offline and then executed with fixed architectures and operating points. Surveys of on-device AI emphasize this "optimize-then-freeze" workflow as the default, with runtime adaptation typically confined to narrow mechanisms or handcrafted heuristics [1].

We adopt an operational definition of Edge AI as *learning-enabled systems deployed on (or near) resource-constrained devices that must deliver utility over time while interacting with the physical world under time-varying budgets* [1]. Under this definition, a non-adaptive deployment corresponds to a fixed operating point - a frozen configuration and execution policy - and is therefore a limiting case: it may remain adequate over short horizons or tightly controlled regimes (e.g., fixed industrial inspection with stable illumination and no thermal variation), but it is operationally incomplete for sustained operation under realistic environmental and operational variability.

Three empirical properties motivate treating Edge AI and Adaptive Edge AI as equivalent operational concepts.

(P1) Non-stationarity is the default. Edge data streams arise from physics and human behavior rather than curated i.i.d. datasets. Even when the semantic task is stable, the effective test distribution evolves (context shifts, illumination/weather changes, motion patterns, sensor drift), so the meaning of "in-distribution" changes over time. This deployment mismatch is the explicit target of recent test-time adaptation work and on-device benchmarks [3], [4].

(P2) Budgets are time-varying and multi-dimensional. Latency, energy, thermal headroom, bandwidth, and privacy constraints fluctuate with device state and user context. A fixed operating point (model, precision, compute budget) cannot remain simultaneously feasible and efficient across regimes. Thermal dynamics and throttling make sustained performance a moving target on mobile/edge hardware [6], and energy efficiency is now treated as a first-order benchmarking dimension across deployment scales [5].

(P3) Interaction induces action-dependent streams. In many edge applications, the system outputs influence subsequent inputs through sensing/control decisions and user interaction. Robots and wearables change what they observe by moving or sampling; monitoring systems trigger interventions that alter the measured process. The resulting stream is partly endogenous to the deployed behavior, so i.i.d. assumptions can fail even in otherwise stationary environments [7].

Together, (P1)-(P3) imply that Edge AI requires an operational capability to *detect* meaningful change, *select* feasible reconfigurations under current budgets, and *apply* them safely over time. We refer to this capability as *adaptivity*; consequently, Edge AI and Adaptive Edge AI coincide as operational concepts. The next section formalizes the Adaptive Edge AI framework and motivates a decade-scale research agenda.

III. The Agent-System-Environment (ASE) lens

Section II motivates Edge AI as long-horizon operation under drift and time-varying budgets, and highlights that progress is hard to compare without time-aware protocols. To formalize adaptivity, we introduce an Agent-System-Environment (ASE) lens. ASE fixes the *state variables* evolving over time - the observation and budget stream, the system configuration/state, and admissible reconfigurations - so that methods can be described and evaluated on common ground.

A. The ASE loop (variables and roles)

At time t, the environment yields observations o_t and operating conditions/budgets c_t (e.g., latency, energy, privacy, connectivity, thermal state). The deployed system is in state s_t (configuration, calibration, memory, and possibly parameters) and produces an output $\hat{y}_t$. An adaptation mechanism selects an action a_t that may change the system state, yielding s_{t+1}:

$$(o_t, c_t) \;\mapsto\; \hat{y}_t = f_{s_t}(o_t), \tag{1}$$

$$a_t = \rho(o_t, s_t, c_t), \qquad s_{t+1} = g(s_t, a_t). \tag{2}$$

Environment (E). The data-generating process and operating regime: context shifts, sensor drift, workload/concurrency, connectivity/privacy state, and hardware conditions (temperature, aging, faults).

System (S). The deployed artifact implementing f_{s_t}: model(s), memory/caches, calibration state, and runtime configuration (precision, routing, offload), including any fallback behavior.

Adaptation mechanism (A). The rule ρ selecting reconfigurations over time (from thresholds to learned decision rules). Actions a_t span execution-level reconfiguration (conditional routing/early exits, precision scaling) and state-level updates (cache/memory refresh, recalibration, selective adaptor/parameter updates, offload/splitting) [2], [8].

The ASE loop can be viewed as a constrained POMDP specialized to the edge setting: state, action, and observation spaces are instantiated around device-level reconfigurations and time-varying hardware and connectivity budgets, fixing a concrete deployment vocabulary without requiring the full generality and computational requirements of unconstrained POMDPs.

Learning under non-stationarity with tight time and memory constraints and delayed or missing labels is well-studied in data stream learning [9], [10]. ASE subsumes this as a special case (fixed s_t, actions limited to parameter updates) and extends it with a multi-dimensional constraint stream c_t and a rich action space a_t that includes execution-level reconfigurations (routing, precision, offload) alongside model updates.

B. Net utility under budgets (execution + reconfiguration)

ASE makes explicit that adaptivity should be evaluated as a *net* gain under budgets: it consumes the same resource budget as inference. Over a horizon T, a generic constrained objective reads $\max_\rho \;\mathbb{E}\Big[\sum_{t=1}^{T} u(\hat{y}_t, y_t) - \lambda \cdot \mathcal{C}_{\text{total}}(s_t, a_t, c_t)\Big]$, subject to time-varying constraint violations $\mathbb{E}\Big[\sum_{t=1}^{T} \text{viol}(c_t)\Big] \le \epsilon_t$, where labels y_t may be delayed or missing.

We separate the cost of running the current configuration from the overhead of changing it:

$$\mathcal{C}_{\text{total}} = \underbrace{\mathcal{C}_{\text{run}}(s_t, c_t)}_{\text{execution}} + \underbrace{\mathcal{C}_{\text{chg}}(s_t, a_t, c_t)}_{\text{reconfiguration overhead}} . \tag{3}$$

TABLE I
TYPICAL INSTANTIATIONS OF ASE VARIABLES IN ADAPTIVE EDGE SYSTEMS: OBSERVABLES $(o_t, c_t) \rightarrow$ ACTIONS $a_t \rightarrow$ FAILURE MODES (UTILITY LOSS / VIOLATIONS OVER TIME).

Observables (o_t, c_t)	**Actions** a_t **(reconfigure** s_t**)**	**Failure modes (over time)**
Uncertainty / calibration drift	Exit depth, compute budget, model selection, abstention	Overconfidence, silent failures, missed escalations
Telemetry (latency/energy/thermal)	Precision scaling, caching, routing, structured sparsity	Budget overruns, instability, tail degradation
Shift/drift indicators (OOD, change tests)	Recalibration, memory refresh, selective updates	False alarms, oscillations, forgetting
Fault/anomaly indicators (sensor/hardware)	Module isolation, redundancy, fallback modes	Misattribution, cascading failures
Connectivity / privacy state	Local-only adaptation, offload/split execution, communication scheduling	Leakage, brittle offline behavior, stale personalization

This separation is operationally relevant in on-device settings where adaptation is periodic, resource-limited, and measured end-to-end [4], and it aligns with benchmarking efforts that emphasize standardized energy accounting to ensure cross-platform comparability [5].

Constraint violations (illustrative). One simple instantiation is a sum of budget exceedances: $\mathrm{viol}(c_t) = \mathbb{1}[\mathrm{lat}_t > b_t^{\mathrm{lat}}] + \mathbb{1}[\mathrm{en}_t > b_t^{\mathrm{en}}] + \mathbb{1}[\mathrm{bw}_t > b_t^{\mathrm{bw}}] + \mathbb{1}[\mathrm{priv}_t < b_t^{\mathrm{priv}}]$, where budgets b_t may vary over time with device state and context.

C. Instantiating (o_t, c_t) and a_t in practice

In deployed edge systems, the stream (o_t, c_t) in (1)-(2) is realized through data-side observables (features, uncertainty, calibration error), and system-side telemetry (latency/energy/thermal state, connectivity/privacy regime), while a_t corresponds to a restricted set of admissible reconfigurations that modify s_t under budgets. Table I summarizes common instantiations of these quantities and the corresponding failure modes that must be controlled over time. With (o_t, c_t, s_t, a_t) explicit, the open problems in Section IV can now be stated as requirements on observability, admissible actions, and long-horizon control of violations [1], [2], [4].

IV. TEN CHALLENGES FOR THE NEXT DECADE

We outline ten algorithmic and systems challenges that follow from the ASE view of Edge AI as operation over time under drift and constraints. Each challenge is stated as: Problem → Core difficulty → Evaluation protocol → Success.

C1: Can we provide theoretical guarantees for adaptive edge systems under drift?

Research question. Can we provide asymptotic and non-asymptotic theoretical guarantees on *risk* and *constraint violations* when the deployed configuration s_t (and possibly parameters) evolves online under non-stationarity?

Problem. Develop guarantees for edge operation over time that jointly cover: (i) risk control under shift (e.g., PAC/PAC-Bayes-style bounds and domain-adaptation bounds) [11]–[13] (ii) non-stationary online-learning guarantees with explicit drift dependence (dynamic/adaptive/interval regret) [14]–[16], and (iii) constraint-aware guarantees that bound budget violations over time (latency/energy/safety/privacy) [17], [18].

Core difficulty. In Edge AI, the learning problem is not only non-i.i.d.; it is often *policy-coupled*: reconfiguration actions a_t (routing, sampling, offload, abstention) affect what is observed and which feedback becomes available, yielding action-dependent data and delayed/partial supervision. As a consequence, both the loss process and the information structure evolve with (s_t, a_t), not only with exogenous drift [19].

Evaluation protocol. Report risk and constraint violations as *time series* over long horizons, explicitly marking drift and update windows. In addition to global averages, include worst-interval / worst-slice summaries aligned with non-stationary regret notions [16]. When uncertainty sets are used to trigger or accept updates, report their online validity under shift (e.g., coverage/control via online conformal prediction) [20].

Success. Guarantees with explicit dependence on drift/variation and constraint models (e.g., variation budgets, path length, long-term constraints), together with empirical evidence of controlled risk and bounded violation rates under declared drift/intervention scripts.

C2: Can an edge system switch paradigms without breaking?

Research question. Can we switch (or hybridize) *learned* and *structured/physics- or rule-based* components online - i.e., change the model class inside s_t via actions a_t - while keeping utility and budget compliance predictable over time?

Problem. Enable reconfiguration actions a_t that modify the system state s_t not only in *execution* (routing, depth, precision), but also in the *computational model* itself: selecting among learned modules, structured estimators/controllers, physics- or rule-based components, or hybrid compositions, depending on (o_t, c_t) (e.g., label rate, physics mismatch, connectivity, safety regime) [2], [4], [21].

Core difficulty. Paradigm switches are discrete. A transition changes both the realized mapping f_{s_t} and the internal state carried across time (calibration, latent/memory state, estimator/controller state). Without an explicit *state-transfer* specification (e.g., $s_{t+1} \leftarrow \mathrm{T}_{a_t}(s_t, o_t, c_t)$) and a switching cost model $\mathcal{C}_{\mathrm{chg}}(s_t, a_t, c_t)$, systems may exhibit transient violations (latency/energy spikes, unsafe outputs) or *chattering* (oscillatory switching) under noisy observables and tight budgets.

Evaluation protocol. Use benchmark streams with declared *mode-change events* in (o_t, c_t): budget shifts, connectivity

drops, changes in label availability, and physics-mismatch regimes. Report: (i) utility-cost trajectories before/during/after each switch; (ii) time-to-recover (return-to-spec time) and failure probability per switch; (iii) constraint violations in the transition window; (iv) switching frequency and hysteresis (to quantify chattering). When switching is gated by uncertainty or exit criteria, include risk-controlled gating variants [8]. For edge settings with split/offload options, include benchmarks where switching couples with placement decisions [22].

Success. Mode changes with bounded transient degradation and bounded violation probability, stable long-horizon behavior (no chattering), and reproducible trade-offs across operating conditions and devices.

C3: When utility drops, can we localize the cause and fix only what is broken?

Research question. Given the stream (o_t, c_t) and the current system state s_t, can an edge system infer *why* utility or constraint compliance degraded (sensor fault, regime shift, component mismatch, or budget shift), and then choose the *smallest effective* action a_t that restores utility while keeping violations bounded?

Problem. Move from anomaly *detection* to anomaly *attribution* and *targeted repair*. Anomaly detection on streams is now well-studied and benchmarked [23], [24], but adaptive Edge AI additionally needs: (i) identifying the fault source (which part of E or S changed), and (ii) selecting a localized intervention (recalibration, memory refresh, module swap, limited parameter/adaptor update) rather than global retraining.

Core difficulty. Attribution is ill-posed under partial observability: multiple causes can co-occur, anomalies propagate across variables and modules, and the same drop in $u(\hat{y}_t, y_t)$ (or rise in $\mathrm{viol}(c_t)$) can be explained by different latent changes. Moreover, interventions themselves can confound diagnosis by altering the subsequent stream. Recent work on temporal/causal root-cause analysis makes progress on structured attribution [25], [26], but integrating attribution with *minimal* interventions that are edge-feasible (global retrain is often infeasible on-device) remains largely open.

Evaluation protocol. Use long-horizon streams with *labeled fault sources* and controlled injections (sensor bias/drift/dropout, packet loss, constraint-regime switches, component failures). Report: (i) detection delay; (ii) attribution precision/recall over fault sources; (iii) *intervention minimality* (changed parameters/modules; energy/time) and *net* regained utility under budgets; (iv) collateral effects on unaffected slices. Where repair is claimed, include minimal-change baselines and (when possible) verifiable post-repair properties [27].

Success. Near-real-time recovery through *localized* actions whose effect is attributable and stable over time: bounded violations, predictable side-effects, and measurable net gain (utility regained minus intervention cost), without frequent full retraining or global re-optimization, aligning with minimal-change principles such as the ones studied in neural network repair/model update methods [28].

C4: Can we make "spend more compute only when necessary" provably safe at the edge?

Research question. Can we design a two-tier (or multi-tier) system where a cheap pathway handles most inputs, and an expensive pathway is invoked only when needed, so that the resulting policy a_t (escalate or not) yields predictable *tail-risk* reduction under drift while respecting time-varying budgets c_t?

Problem. Construct *anytime* systems by decomposing the deployed state s_t into fast and slow components ("System 1" and "System 2"), and enabling actions a_t that allocate computation on demand (e.g., early exits, conditional routing, selective activation) [2], [29]. The goal is not lower average FLOPs per se, but reliability under constraints: spend additional compute only on inputs and regimes where it measurably reduces risk.

Core difficulty. Escalation is simultaneously a *reliability* decision and a *budget* decision. Under shift, naive difficulty/uncertainty scores can become miscalibrated, causing missed escalations precisely on hard/OOD time intervals; under tight budgets, overly conservative gating collapses into frequent escalations and erases savings. Technically, the challenge is to produce uncertainty signals that remain meaningful. Recent work connects early exiting to distribution-free risk control and to *nested* (sequentially consistent) prediction sets, enabling controlled escalation policies [8], [30].

Evaluation protocol. Report performance as *risk-cost tradeoffs* over time. At minimum include: (i) tail-risk at fixed *average* latency/energy; (ii) escalation frequency and escalation errors (missed vs unnecessary escalations); (iii) constraint violations during drift regimes (time series, not only steady state). When risk control is claimed, report risk-coverage style curves and risk-controlled operating points rather than only accuracy-FLOPs plots [8], [31].

Success. Stable long-horizon reduction in tail risk with bounded violation rates, achieved through escalation policies that remain calibrated under drift and do not exhibit chattering or regime-dependent pathologies.

C5: Can we compose and update modular edge intelligence without breaking the whole system?

Research question. Can we represent the deployed state s_t as a *set of composable modules* - including *memory* - that can be added or updated locally (via actions a_t) while preserving predictable system-level behavior and budget compliance?

Problem. Move from monolithic models toward modular edge systems: libraries of experts/adapters/skills plus explicit memory components (episodic buffers, retrieval indices, caches) that support long-horizon operation and context specialization. In this view, s_t contains multiple interacting subsystems (compute modules and memory state), and adaptation actions a_t include adding/removing modules, updating a single module, or refreshing/compacting memory, rather than retraining the full stack [32]–[34]. This parallels the functional motivation for specialization in biological systems (distinct subsystems for rapid recall vs slower consolidation).

Core difficulty. Changing one module can shift hidden representations, calibration, routing decisions, or retrieval dis-

tributions.Memory exacerbates this: retrieval-based behavior is distribution-dependent, and local changes to the embedding/key space can invalidate stored items. Therefore, modular updates require explicit *compatibility mechanisms* and lightweight verification to prevent local fixes that degrade global utility or violate constraints.

Evaluation protocol. Use *module-isolation* tests aligned with ASE: apply actions that modify exactly one component of s_t (swap/update a single module, add a new skill, or refresh/compact memory) while freezing the rest. Report: (i) system-level utility and constraint violations over time; (ii) changes in performance metrics on unaffected tasks/slices (including calibration and abstention behavior); (iii) module-level accountability (attribution of gains/regressions to the modified component); (iv) memory-specific metrics when present (retrieval hit-rate / staleness, memory growth, and energy/time per memory maintenance step).

Success. Local updates with predictable global effects: improvements persist, deteriorations are bounded and detectable, and modules (including memory representations) remain reusable across contexts.

C6: Can we trigger, accept, or rollback adaptation without ground truth?

Research question. When labels are delayed, sparse, or absent, can we decide *when* to adapt and *whether* an update should be kept (or rolled back), while controlling risk and constraint violations over time?

Problem. In the ASE loop, supervision is part of the environment: y_t may arrive late, intermittently, or under a fixed labeling budget. The system must therefore solve two coupled decisions: (i) *triggering* (choose a_t that initiates an adaptation step), and (ii) *acceptance* (keep/rollback a modified state s_{t+1}). This includes edge-feasible mechanisms that *change the feedback process* such as selective querying (active learning) or deferral, and mechanisms that *substitute* supervision with proxy objectives (self-/semi-supervised signals) when labels are unavailable.

Core difficulty. Without ground truth, acceptance relies on proxy signals (self-supervised losses, entropy, reconstruction error, agreement across exits/models) that can become miscalibrated under drift. Moreover, unsupervised test-time adaptation can create feedback loops (confirmation bias) that silently degrade reliability, so "improvement" cannot be assumed from decreasing proxy losses alone [3]. Active querying helps but is itself constrained (cost, privacy, latency, user burden) and must be allocated where it most reduces risk.

Evaluation protocol. Use streams with explicit label-delay processes and/or fixed label budgets; include conditions where only a small calibration window is available. Report: (i) risk calibration under drift; (ii) false-trigger rate (adapt when unnecessary) and miss rate (fail to adapt when needed); (iii) acceptance errors (keep a harmful update vs rollback a beneficial one) and the cost of rollback; (iv) utility and constraint violations as *time series* across drift/update windows. Include abstention/selective-prediction baselines as safety mechanisms under shift [35], [36].

Success. Update-trigger and acceptance criteria with finite-sample, distribution-free (finite-sample) control of risk/violations from small calibration windows, remaining valid in online/non-stationary settings - e.g., via conformal risk control and online conformal inference [20], [37].

C7: Is efficient inference *irrelevant if maintenance dominates?*

Research question. Over a long horizon, can we minimize the *total* cost of operating an adaptive edge system $\sum_t \mathcal{C}_{\text{run}}(s_t, c_t) + \mathcal{C}_{\text{chg}}(s_t, a_t, c_t)$ while keeping utility above specification and constraint violations bounded?

Problem. Shift the objective of *efficiency* from static inference FLOPs to *lifecycle-efficient operation*: training, deployment, monitoring, periodic adaptation/maintenance, and (when relevant) communication. In ASE terms, this is precisely the cost term in (3): the system must be efficient at executing f_{s_t} and also at deciding and applying changes to s_t over time.

Core difficulty. In realistic deployments, the dominant cost can move from execution to *maintenance*: monitoring, periodic adaptation/personalization, and distributed update mechanisms can exceed inference energy/time over the horizon. Moreover, energy/latency accounting is platform-dependent and easy to misreport without standardized methodology, making comparisons across heterogeneous hardware and operating states unreliable [5], [38]. This is amplified on-device, where background load, thermal throttling, and OS scheduling affect both $\mathcal{C}_{\text{run}}$ and $\mathcal{C}_{\text{chg}}$ [1].

Evaluation protocol. Report *horizon-level* metrics: total energy/time to keep utility above a target (or within a risk bound) *and* bounded tail risk, including monitoring and update costs. Disclose measurement methodology (power logging source, integration window, device state, thermal regime) for reproducibility [5]. In addition to totals, report an amortized metric such as Joules per unit of maintained/recovered utility over the horizon, and separate $\sum_t \mathcal{C}_{\text{run}}$ from $\sum_t \mathcal{C}_{\text{chg}}$.

Success. Methods and systems that achieve lower lifecycle energy/time with stable long-horizon behavior, demonstrated on resource-constrained devices, with explicit separation of execution vs maintenance costs and clear evidence that the net gain persists under realistic operating conditions [1], [39].

C8: Can we adapt privately when the network is unreliable?

Research question. When privacy and connectivity are part of the constraint stream c_t, can an edge system personalize and adapt using only admissible observables and actions, while remaining stable through extended offline periods?

Problem. Enable personalization and continual adaptation when raw data cannot leave the device and coordination is intermittent. In ASE terms, privacy and connectivity constrain both the information available for learning (what feedback can be used or shared) and the action space (which updates are permissible and when synchronization can occur), favoring local-only updates (e.g., adapters) with intermittent aggregation when communication is available [40], [41].

Core difficulty. Privacy limits what statistics/representations/gradients can be communicated; connectivity induces asynchronous, bursty synchronization. The system must therefore (i) avoid unstable personalization during long offline intervals (overfitting/forgetting), (ii) prevent leakage through updates or stored state, and (iii) remain robust to staleness and heterogeneity when aggregation resumes [40], [41]. These constraints are coupled: more aggressive local adaptation increases drift from shared baselines and complicates safe re-alignment.

Evaluation protocol. Use realistic connectivity traces (online/offline schedules) and explicit communication budgets. Report: (i) utility vs communication (bytes/rounds) and local compute/energy; (ii) decay and recovery across offline intervals (time series); (iii) stability under heterogeneity when synchronization resumes; (iv) privacy reporting consistent with the threat model (e.g., DP accounting if claimed, or explicit attack evaluation otherwise). Include baselines separating local-only adaptation from intermittent coordination [40].

Success. Competitive personalization with minimal communication and stable long-horizon behavior under realistic offline durations, with privacy guarantees (or empirically validated protections) under an explicit threat model.

C9: Can we keep guarantees when the hardware becomes part of the drift?

Research question. When the compute substrate and sensing stack evolve over time, can we maintain predictable utility and bounded violations by adapting s_t (and selecting mitigation actions a_t) under tight budgets?

Problem. Maintain utility and constraint compliance when the *compute substrate itself* changes with conditions and aging, including thermal effects (temperature dependence, DVFS/throttling), voltage-margin trimming/undervolting, soft errors in memory/compute, wear-out and drift, and non-idealities in emerging accelerators (e.g., analog noise and conductance drift in in-memory compute) [6], [42]–[44]. In ASE terms, this is a coupled drift: hardware affects the realized mapping f_{s_t}, the constraint stream c_t (latency/energy regimes), and sometimes the observation stream o_t (sensor/ADC drift).

Core difficulty. Hardware variability induces *state-dependent*: the same s_t and o_t can produce different outputs depending on temperature/voltage/aging state, breaking the assumption that the deployed model is a fixed function plus i.i.d. noise. Failures can be silent (Silent Data Corruption, SDC), interact with conditional execution (different routes stress different units), and occur precisely when budgets are tight, while edge constraints limit frequent recalibration; robust operation therefore couples learning, calibration, and device/accelerator characterization [45], [46].

Evaluation protocol. Run long-horizon stress tests that sweep temperature/voltage/frequency, induce throttling, and inject controlled faults/drift in memory and compute. Report: (i) utility trajectories and tail-risk, including SDC rate where applicable; (ii) constraint violations under throttling and degraded regimes; (iii) robustness under specified fault/drift injection models; (iv) frequency and *cost* (energy/time) of mitigation actions (error detection, redundancy, recalibration/compensation) [45], [47]. When analog/IMC is involved, include drift profiles and compensation overhead [44].

Success. Sustained utility with bounded violations under realistic variability and degradation, with low overhead and predictable degradation modes: lightweight compensation that tracks substrate drift (and its induced c_t changes) without frequent full retraining [48], [49].

C10: What does it mean to win *when systems adapt over time?*

Research question. Which protocols and metrics make adaptive edge systems comparable when (o_t, c_t), s_t, and permissible actions a_t evolve over time?

Problem. Static test sets and single-shot reporting cannot characterize adaptive behavior: the relevant phenomena are *transient* and *path-dependent* - recovery after drift, oscillations under repeated interventions, and time-varying budget violations. In ASE terms, *performance* is not a scalar but a trajectory of utility and constraint satisfaction under a declared stream and adaptation rules.

Core difficulty. Reported gains depend strongly on choices that are often under-specified: (i) the *shift script* (type/severity/timing of changes in (o_t, c_t)), (ii) the *adaptation schedule* (continuous vs periodic; what data/labels are available and when), and (iii) the *cost accounting* (what is included in energy/latency and how it is measured on-device). Recent on-device adaptation benchmarks emphasize that these details can dominate conclusions and must be reported explicitly [4], [50], while system-level benchmarking efforts stress standardized measurement methodology to make cross-platform comparisons meaningful [5].

Evaluation protocol. Evaluate *trajectories over time*, not only the post-adaptation steady state. Report: (i) utility $U(t)$ (including calibration measures when relevant), (ii) cost $\mathcal{C}_{\text{run}}(t)$ and $\mathcal{C}_{\text{chg}}(t)$ (execution vs reconfiguration), and (iii) violation indicators $V(t)$ derived from $\text{viol}(c_t)$.

Include recovery and stability metrics, e.g.: (i) *Energy-to-Recover (E2R)*: Joules from shift onset (or detection) to regain within δ of pre-shift utility; (ii) *Time-to-Recover (T2R)*: time/frames to recover to the same threshold; (iii) *Stability Score (SS)*: oscillation/variance of $U(t)$ and $V(t)$ under repeated shifts and interventions. When reporting energy/latency, disclose the measurement protocol (instrumentation, integration window, phases included, device state/thermal regime), following reproducible benchmarking best practices [5], [51].

Success. Community benchmarks and reporting standards that fix: (i) declared shift/intervention scripts, (ii) recovery/stability metrics, and (iii) on-device cost accounting - so that adaptive edge systems are comparable across algorithms, hardware, and operating regimes [4], [5], [50].

Reporting standard for adaptive Edge AI: Challenge C10 calls for comparability under drift and intervention scripts and explicit on-device cost reporting. As a concrete step, we propose the *reporting standard* in Table II: it specifies the drift and constraints regime, the admissible adaptation actions,

TABLE II
MINIMAL CHECKLIST FOR ADAPTIVE EDGE AI EXPERIMENTS (TO ENABLE COMPARABILITY).

Report (at minimum):
(D) Drift/interventions: what changes (data/task/hardware/user), severity/timescale, and the declared shift/intervention (change points known/unknown, staged/gradual, injected faults).
(C) Constraints: latency/energy/memory/privacy/connectivity budgets and how c_t varies over time (device state, throttling, offline periods).
(A) Actions: admissible a_t (routing, precision, memory, params, offload), adaptation schedule (continuous/periodic), rollback/fallback policy.
(S) Signals: observables used to trigger/select actions (uncertainty, drift tests, telemetry) and their thresholds/costs.
(M) Metrics as time series: $U(t)$, $\mathcal{C}_{\text{run}}(t)$, $\mathcal{C}_{\text{chg}}(t)$, and $V(t)$; include recovery/stability summaries when relevant.
(P) Protocol: time-aware splits; online vs offline; label availability and delay/budget model; data used for adaptation vs evaluation.
(B) Baselines: frozen deployment; oracle/upper bounds where meaningful; ablations isolating each action type.

and the time-series metrics required to interpret results beyond static accuracy.

V. CONCLUSION

Edge AI requires long-horizon operation in the wild: the data stream, compute budgets, connectivity, and even the compute substrate vary over time. Under these conditions, a fixed deployment checkpoint inevitably drifts toward either *utility failure* (miscalibration and risk growth under shift) or *budget failure* (latency or energy violations under changing device state), often with silent degradation. This motivates our operational thesis that Edge AI and Adaptive Edge AI coincide in practice.

The ASE lens proposed in this paper makes this thesis precise by exposing the minimal objects that determine adaptive behavior over time: the observable stream (o_t, c_t), the evolving system state s_t, and the admissible interventions a_t under constraints. Within this coordinate system, we formulate ten research questions for the next decade - from guarantees under drift to modularity, private/offline adaptation, hardware variability, and time-series evaluation protocols - and we propose a minimal reporting standard to make progress comparable across algorithms, devices, and operating regimes.

ACKNOWLEDGMENTS

This paper is supported by PNRR-PE-AI FAIR project funded by the NextGeneration EU program.

REFERENCES

[1] X. Wang, Z. Tang, J. Guo, T. Meng, C. Wang, T. Wang, and W. Jia, "Empowering edge intelligence: A comprehensive survey on on-device ai models," *ACM Comput. Surv.*, vol. 57, no. 9, Apr. 2025. [Online]. Available: https://doi.org/10.1145/3724420

[2] S. Scardapane, A. Baiocchi, A. Devoto, V. Marsocci, P. Minervini, and J. Pomponi, "Conditional computation in neural networks: Principles and research trends," pp. 175–190, 2024. [Online]. Available: https://journals.sagepub.com/doi/abs/10.3233/IA-240035

[3] Z. Xiao and C. G. M. Snoek, "Beyond model adaptation at test time: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2411.03687

[4] M. Danilowski, S. Chatterjee, and A. Ghosh, "Botta: Benchmarking on-device test time adaptation," 2025. [Online]. Available: https://arxiv.org/abs/2504.10149

[5] A. Tschand, A. T. R. Rajan, S. Idgunji, A. Ghosh, J. Holleman, C. Kiraly, P. Ambalkar, R. Borkar, R. Chukka, T. Cockrell, O. Curtis, G. Fursin, M. Hodak, H. Kassa, A. Lokhmotov, D. Miskovic, Y. Pan, M. P. Manmathan, L. Raymond, T. S. John, A. Suresh, R. Taubitz, S. Zhan, S. Wasson, D. Kanter, and V. J. Reddi, "Mlperf power: Benchmarking the energy efficiency of machine learning systems from microwatts to megawatts for sustainable ai," 2025. [Online]. Available: https://arxiv.org/abs/2410.12032

[6] T. Tan and G. Cao, "Thermal-aware scheduling for deep learning on mobile devices with npu," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 10 706–10 719, 2024.

[7] D. Mansfield and A. Montazeri, "A survey on autonomous environmental monitoring approaches: towards unifying active sensing and reinforcement learning," *Frontiers in Robotics and AI*, vol. Volume 11 - 2024, 2024. [Online]. Available: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2024.1336612

[8] M. Jazbec, A. Timans, T. H. Veljković, K. Sakmann, D. Zhang, C. A. Naesseth, and E. Nalisnick, "Fast yet safe: Early-exiting with risk control," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 129 825–129 854. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/ea5a63f7ddb82e58623693fd1f4933f7-Paper-Conference.pdf

[9] J. a. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, Mar. 2014. [Online]. Available: https://doi.org/10.1145/2523813

[10] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: A review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, pp. 1261–1274, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217315928

[11] M. Seeger, "Pac-bayesian generalisation error bounds for gaussian process classification," *J. Mach. Learn. Res.*, vol. 3, no. null, p. 233–269, Mar. 2003. [Online]. Available: https://doi.org/10.1162/153244303765208386

[12] M. Hennequin, K. Benabdeslem, and H. Elghazel, "Pac-bayesian domain adaptation bounds for multi-view learning," 2024. [Online]. Available: https://arxiv.org/abs/2401.01048

[13] Z. Wang and Y. Mao, "On f-divergence principled domain adaptation: An improved framework," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 6711–6748. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/0ccd06ff26fd6a7829293ce90e0e7f7d-Paper-Conference.pdf

[14] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ser. ICML'03. AAAI Press, 2003, p. 928–935.

[15] E. Hazan and C. Seshadhri, "Efficient learning algorithms for changing environments," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 393–400. [Online]. Available: https://doi.org/10.1145/1553374.1553425

[16] P. Zhao, Y.-F. Xie, L. Zhang, and Z.-H. Zhou, "Efficient methods for non-stationary online learning," *Journal of Machine Learning Research*, vol. 26, no. 208, pp. 1–66, 2025. [Online]. Available: http://jmlr.org/papers/v26/23-1188.html

[17] S. Hutchinson and M. Alizadeh, "Safe online convex optimization with multi-point feedback," in *Proceedings of the 6th Annual Learning for Dynamics &; Control Conference*, ser. Proceedings of Machine Learning Research, A. Abate, M. Cannon, K. Margellos, and A. Papachristodoulou, Eds., vol. 242. PMLR, 15–17 Jul 2024, pp. 168–180. [Online]. Available: https://proceedings.mlr.press/v242/hutchinson24a.html

[18] A. Lechowicz, N. Christianson, B. Sun, N. Bashir, M. Hajiesmaili, A. Wierman, and P. Shenoy, "Chasing convex functions with long-term constraints," in *Proceedings of the 41st International*

Conference on Machine Learning, ser. Proceedings of Machine Learning Research, R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, Eds., vol. 235. PMLR, 21–27 Jul 2024, pp. 26 259–26 289. [Online]. Available: https://proceedings.mlr.press/v235/lechowicz24a.html

[19] J. Suk, "Adaptive smooth nonstationary bandits," *SIAM Journal on Mathematics of Data Science*, vol. 7, no. 3, pp. 1265–1291, 2025. [Online]. Available: https://doi.org/10.1137/24M167651X

[20] I. Gibbs and E. J. Candès, "Conformal inference for online prediction with arbitrary distribution shifts," *Journal of Machine Learning Research*, vol. 25, no. 162, pp. 1–36, 2024. [Online]. Available: http://jmlr.org/papers/v25/22-1218.html

[21] A. Quarteroni, P. Gervasio, and F. Regazzoni, "Combining physics-based and data-driven models: advancing the frontiers of research with scientific machine learning," *Mathematical Models and Methods in Applied Sciences*, vol. 35, no. 04, pp. 905–1071, 2025. [Online]. Available: https://doi.org/10.1142/S0218202525500125

[22] M. Colocrese, E. Koyuncu, and H. Seferoglu, "Early-exit meets model-distributed inference at edge networks," 2024. [Online]. Available: https://arxiv.org/abs/2408.05247

[23] Q. Liu and J. Paparrizos, "The elephant in the room: Towards a reliable time-series anomaly detection benchmark," in *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. [Online]. Available: https://openreview.net/forum?id=R6kJtWsTGy

[24] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ACM Comput. Surv.*, vol. 57, no. 1, Oct. 2024. [Online]. Available: https://doi.org/10.1145/3691338

[25] J. Rehak, S. Youssef, and J. Beyerer, "Root cause analysis using anomaly detection and temporal informed causal graphs," 2024. [Online]. Available: https://publica.fraunhofer.de/handle/publica/467932

[26] X. Han, S. Absar, L. Zhang, and S. Yuan, "Root cause analysis of anomalies in multivariate time series through granger causal discovery," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=k38Th3x4d9

[27] F. Fu, Z. Wang, W. Zhou, Y. Wang, J. Fan, C. Huang, Q. Zhu, X. Chen, and W. Li, "Reglo: Provable neural network repair for global robustness properties," vol. 38, no. 11, Mar. 2024, pp. 12 061–12 071. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/29094

[28] Z. Chen, J. Zhou, Y. Sun, J. Wang, Q. Xuan, and X. Yang, "Interpretability based neural network repair," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2024. New York, NY, USA: Association for Computing Machinery, 2024, p. 908–919. [Online]. Available: https://doi.org/10.1145/3650212.3680330

[29] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2464–2469.

[30] M. Jazbec, P. Forré, S. Mandt, D. Zhang, and E. Nalisnick, "Early-exit neural networks with nested prediction sets," in *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, ser. Proceedings of Machine Learning Research, N. Kiyavash and J. M. Mooij, Eds., vol. 244. PMLR, 15–19 Jul 2024, pp. 1780–1796. [Online]. Available: https://proceedings.mlr.press/v244/jazbec24a.html

[31] S. Goren, I. Galil, and R. El-Yaniv, "Hierarchical selective classification," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. [Online]. Available: https://openreview.net/forum?id=wzof7Y66xs

[32] M. Hu, H. Chang, B. Ma, S. Shan, and X. CHEN, "Scalable modular network: A framework for adaptive learning via agreement routing," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=pEKJl5sflp

[33] S. Liu, Y. Yang, X. Li, D. A. Clifton, and B. Ghanem, "Enhancing online continual learning with plug-and-play state space model and class-conditional mixture of discretization," pp. 20 502–20 511, 2025.

[34] T. Liu, J. Li, Y. Zheng, H. Niu, Y. Lan, X. Xu, and X. Zhan, "Skill expansion and composition in parameter space," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=GLWf2fq0bX

[35] H. Liang, L. Peng, and J. Sun, "Selective classification under distribution shifts," *Transactions on Machine Learning Research*, 2024. [Online]. Available: https://openreview.net/forum?id=dmxMGW6J7N

[36] A. Pugnana, L. Perini, J. Davis, and S. Ruggieri, "Deep neural network benchmarks for selective classification," *Journal of Data-centric Machine Learning Research*, 2024, reproducibility Certification. [Online]. Available: https://openreview.net/forum?id=xDPzHbtAEs

[37] A. N. Angelopoulos, S. Bates, A. Fisch, L. Lei, and T. Schuster, "Conformal risk control," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=33XGfHLtZg

[38] E. Barbierato and A. Gatti, "Toward green ai: A methodological survey of the scientific literature," *IEEE Access*, vol. 12, pp. 23 989–24 013, 2024.

[39] S. Li, G. Yuan, Y. Dai, T. Wang, Y. Wu, A. K. Jones, J. Hu, Tony, Geng, Y. Wang, B. Yuan, Y. Ding, and X. Tang, "Redundancy-aware efficient continual learning on edge devices," 2025. [Online]. Available: https://arxiv.org/abs/2401.16694

[40] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. Vincent Poor, "Federated learning for internet of things: A comprehensive survey," pp. 1622–1658, 2021.

[41] Z. Zhong, W. Yuan, L. Qu, T. Chen, H. Wang, X. Zhao, and H. Yin, "Towards on-device personalization: Cloud-device collaborative data augmentation for efficient on-device language model," New York, NY, USA, Jan. 2026. [Online]. Available: https://doi.org/10.1145/3779452

[42] M. H. Ahmadilivani, M. Taheri, J. Raik, M. Daneshtalab, and M. Jenihhin, "A systematic literature review on hardware reliability assessment methods for deep neural networks," *ACM Comput. Surv.*, vol. 56, no. 6, Jan. 2024. [Online]. Available: https://doi.org/10.1145/3638242

[43] C. Bolchini, L. Cassano, and A. Miele, "Resilience of deep learning applications: A systematic literature review of analysis and hardening techniques," *Computer Science Review*, vol. 54, p. 100682, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013724000662

[44] M. J. Rasch, F. Carta, O. Fagbohungbe, and T. Gokmen, "Fast and robust analog in-memory deep neural network training," *Nature Communications*, vol. 15, no. 1, p. 7133, Aug 2024. [Online]. Available: https://doi.org/10.1038/s41467-024-51221-z

[45] G. Yu, G. Tan, H. Huang, Z. Zhang, P. Chen, R. Natella, Z. Zheng, and M. R. Lyu, "A survey on failure analysis and fault injection in ai systems," New York, NY, USA, Dec. 2025. [Online]. Available: https://doi.org/10.1145/3732777

[46] T. Chen, H. Geng, Q. Sun, S. Wan, Y. Sun, H. Yu, and B. Yu, "Wages: The worst transistor aging analysis for large-scale analog integrated circuits via domain generalization," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 29, no. 5, Aug. 2024. [Online]. Available: https://doi.org/10.1145/3659950

[47] W. Zheng, B. Xu, J. Gu, and H. Chen, "Save: software-implemented fault tolerance for model inference against gpu memory bit flips," in *Proceedings of the 2025 USENIX Conference on Usenix Annual Technical Conference*, ser. USENIX ATC '25. USA: USENIX Association, 2025.

[48] P. Katti, C. Ruah, O. Simeone, B. M. Al-Hashimi, and B. Rajendran, "Bayes2imc: In-memory computing for bayesian binary neural networks," pp. 5422–5435, 2025.

[49] M. Martemucci, F. Rummens, Y. Malot, T. Hirtzlin, O. Guille, S. Martin, C. Carabasse, A. F. Vincent, S. Saïghi, L. Grenouillet, D. Querlioz, and E. Vianello, "A ferroelectric–memristor memory for both training and inference," *Nature Electronics*, vol. 8, no. 10, pp. 921–933, Oct 2025. [Online]. Available: https://doi.org/10.1038/s41928-025-01454-7

[50] Z. Fan, Z. Wan, C.-K. Liu, A. Lu, K. Bhardwaj, and A. Raychowdhury, "Benchmarking test-time dnn adaptation at edge with compute-in-memory," *ACM J. Auton. Transport. Syst.*, vol. 1, no. 3, Jul. 2024. [Online]. Available: https://doi.org/10.1145/3665898

[51] P. Bartoli, C. Veronesi, A. Giudici, D. Siorpaes, D. Trojaniello, and F. Zappa, "Benchmarking energy and latency in tinyml: A novel method for resource-constrained ai," pp. 1–8, 2025.