Towards Trustworthy Interactive AI at the Edge

Zexin Li | Nanyang Technological University

My research advances trustworthy interactive AI through predictable execution and verifiable model behavior. Interactive AI, such as voice assistants and real-time world models, must answer a person while the interaction is in progress. On-device execution removes the network round trip but leaves limited memory and compute. For a voice assistant, a correct reply that starts after the conversational turn has passed cannot continue the dialogue. A model that fits during inference may exhaust memory when an online update begins. An optimized executable may run faster yet disagree with the reference model. These failures motivate my central question.

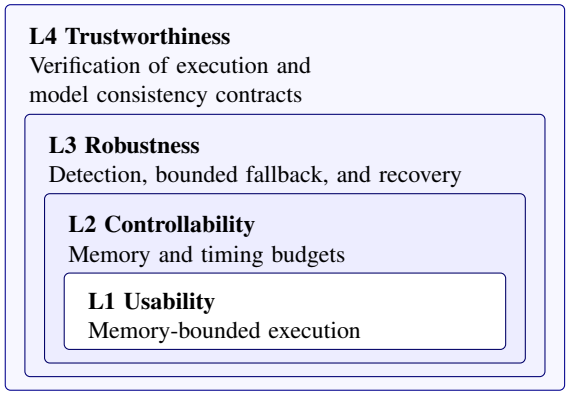
Central Research Question

How can an edge AI system deliver useful intelligence under changing conditions while explicitly defining its operating limits, governing adaptation and recovery, and providing verifiable evidence about its behavior?

The underlying systems problem is that adaptation changes both execution demands and the assumptions supporting correct operation. A smaller model may fit in memory, yet loading it alongside the current model can exhaust capacity or delay an urgent task. Faster compiled code may meet a deadline while disagreeing with its reference model. Resource control and behavioral verification therefore need a common interface: the runtime must know what a configuration requires, which claims its evidence supports, and when a change invalidates those claims. My central research direction is to make these dependencies explicit enough to govern deployment decisions.

I have built foundations on both sides of this interface. In R^3 and AOCL, I formulated the resource-management problems, designed and implemented the runtime mechanisms, and developed their analyses and evaluations [1, 2]. These systems connect learning choices to memory and timing constraints. In ProCerDCL, my co-first-author preprint, I formulated the compiler-consistency problem and developed the system and experimental evaluation, working with my co-first author on a statistically grounded verification procedure [3]. Together, these projects move from controlling whether computation can execute to establishing evidence about the behavior of what executes. The next challenge is to preserve both as a deployed system adapts.

Four nested scopes organize this agenda. L1 establishes memory-bounded usability; L2 adds timing control; L3 encompasses both within fallback and recovery; L4 encloses the system with explicit verification obligations. The figure denotes inclusion, not a sequence of independent stages. A fallback must obey the same resource constraints as normal operation, and a verification claim must account for the policy selecting both. I distinguish measured performance, statistical guarantees, and properties verified within bounded models.



For the voice assistant, switching speech models is a decision about memory, deadlines, recovery, and evidence together. A runtime might have enough space for the destination model but insufficient headroom for the switch; alternatively, execution may remain feasible while the evidence about model consistency no longer applies. These cases require different responses. Treating both as generic overload would hide the reason a deployment can no longer support its claims.

My five-year agenda asks three connected questions: how to admit transitions under joint memory and timing constraints, how to bound recovery on shared resources, and when behavioral evidence can survive a configuration or distribution change. The intended outcome is a runtime contract that developers can inspect and enforce: it identifies admissible changes, the response available when assumptions fail, and the evidence that must be renewed. The goal is to make adaptation accountable throughout the deployment lifecycle.

Research foundation: making learning executable and its behavior assessable

Jointly managing memory and learning deadlines. In R^3 (RTSS 2023), I identified a coupling that treating memory as a fixed capacity misses: batch size and replay-buffer size compete for memory while affecting training time and learning quality differently [1]. I designed a deadline-driven batch controller, a replay-memory organization that reduces redundant storage, and a coordinator that adjusts memory reservations between them. I implemented the runtime and designed its analysis and evaluation. In the reported DDQN experiments across three platforms, both episode- and step-level variants observed zero training-task deadline misses, compared with 54.5–57.9% for MAX-A, the learning library's preset configuration; these are measurements under the study's intermediate-deadline assignments, not a universal zero-miss guarantee. The contribution is to make learning configuration part of resource control: useful learning requires coordinating the allocation of memory with the computation it enables. This insight anchors L1 within L2.

Making concurrent learning responsive to current inference. AOCL, accepted at RTSS 2026, addresses a different bottleneck: serializing learning and inference wastes execution opportunities, but unmanaged overlap can delay the predictions needed for a timely response [2]. I formulated this coordination problem, designed and implemented a runtime that adjusts batch size and alternation intervals using feedback, and developed its timing analysis and experiments. The runtime couples adaptive overlap with a deadline-aware retreat to a feasible configuration. In a case study with input streams under changing illumination, semantic-segmentation training throughput reached $3.1\times$ that of sequential Avalanche, while streaming accuracy changed from 94% to 93%. In separate experiments with concurrent ResNet-20 training and inference, AOCL achieved the lowest or tied-lowest deadline-miss rate in 13 of 15 benchmark/deadline settings. Its feasibility analysis is conditional on execution-cost assumptions; it does not prove zero misses at every adaptive configuration. The broader result is that adaptation progress and immediate responsiveness can be coordinated by the runtime, with their trade-offs made visible.

Connecting efficient execution to evidence about model behavior. ProCerDCL studies a gap that resource control cannot close: compiler optimization can change model decisions, and observing no disagreements on a test set does not establish a bound on unseen inputs [3]. I formulated the problem, implemented the system, and designed the evaluation; my co-first author Weibo Zhang was primarily responsible for establishing the statistical guarantees. Our framework uses a single sequential test to return acceptance, rejection, or an inconclusive verdict with direction-specific error control. In paired one-model clean-label radius searches, the interval-family baseline required 3.6–4.2 times as many unique samples under matched numerical parameters and a weaker decision contract. Separately, two-model consistency checks across 840 input–backend instances from 21 tasks yielded 817 acceptances, 19 inconclusive verdicts, and 4 rejections at a violation threshold of 10^{-3}. These guarantees require a fixed sampling distribution, independent draws, and stateless oracle behavior; they do not establish agreement on every input. This preprint provides a basis for treating behavioral evidence as a deployment resource whose cost and validity must both be managed.

Extending these ideas across learning and interactive systems. Collaborative projects already address workloads in which a person waits for the output: RT-LM manages input-dependent language-model latency, and BOXR targets motion-sensitive XR [4, 5]. Further collaborations cover multi-GPU training/inference scheduling (LeMix, RTSS 2025 Outstanding Paper Award) and energy-aware learning (DuoJoule) [6, 7]. My first-author RED (RTSS 2023) coordinates task dependencies through intermediate deadlines and on-demand synchronization [8]. My co-first-author MixTraining (TMLR 2026) studies the compute–performance trade-off through interleaved self-supervised and supervised learning [9]; my Orion preprint studies memory management for online continual learning [10]. These experiences motivate my next step: connecting workload-specific mechanisms through contracts that remain useful when configurations and operating conditions change.

The common limitation of these foundations is that a result for one workload or configuration does not automatically survive a deployment change. R^3 and AOCL manage execution demands, while ProCerDCL evaluates behavior under fixed assumptions. My future work will connect these capabilities by making changes to those assumptions visible to the runtime. A successful system must determine which obligations a transition preserves and which it must establish again.

Research agenda: contracts for adaptation, recovery, and evidence

Over the next five years, I plan to develop a runtime in which adaptation is governed by explicit execution and evidence obligations. A streaming voice assistant and an action-conditioned real-time world model will provide common testbeds. Three questions define the contracts and decision procedures I will develop.

When is a transition feasible, even if both endpoint configurations are feasible? A model switch can temporarily require two sets of weights, consume memory bandwidth, and block tasks already approaching their deadlines. Building on my resource-control work and prior response-time analysis of processing chains [1, 2, 11], I will develop transition contracts that couple memory lifetimes with bounds on interference and end-to-end response time. The challenge is to admit useful adaptation without assuming that transient costs equal steady-state costs. I will study staged loading, ordered release of state, and admission rules that reserve a feasible fallback throughout a transition. Success requires sound admission under declared cost bounds and fewer unnecessary rejections than independently budgeted memory and timing policies, measured alongside response latency. This extends the unit of scheduling from a task or configuration to a resource-constrained transition.

How can recovery remain bounded when it shares resources with the workload it supervises? Simplex separates advanced and backup control [12]; shared-accelerator AI adds a resource-dependence problem. Here the backup is a degraded but timely response. A backup running on a CPU can still be delayed by memory traffic from ongoing accelerator work. I will develop recovery contracts that account for residual non-preemptive execution, shared-memory interference, and the resources required to stop, hold, and resume. A supervisor will distinguish confidence-driven adaptation from violations of explicit resource limits. Admission must preserve the backup's response budget, and recovery must re-establish the conditions for normal execution before re-entry. Success means a conditional bound from detection to fallback response, validated under adversarial contention, together with useful recovery rather than indefinite degraded operation. Timed-automata analysis [13] will check bounded supervisor models; traces will test whether implementations conform to their assumptions.

Which behavioral evidence survives a change in execution mode or input distribution? ProCerDCL provides distribution-specific evidence for a fixed oracle [3]; adaptive deployment can change both the executable and the inputs it encounters. I will develop evidence contracts that record these dependencies and rules for retaining, invalidating, or renewing each claim. An initial case is a scheduling change that preserves a stateless executable's outputs; harder cases include recompilation, model updates, and distribution drift. I will investigate compositional and statistical conditions for reusing evidence, with explicit error-budget accounting across repeated checks. Unsupported transfer will trigger re-verification or an inconclusive status. Success requires sound reuse under stated conditions and lower verification cost than checking every configuration afresh. Execution proofs and statistical consistency verdicts will remain distinct contract clauses: neither implies correctness of every prediction.

Evaluation and longer-term impact

I will begin with transition admission and reproducible interference measurements, extend the runtime to supervised recovery, and then study evidence reuse across modes and updates. Evaluation will combine component studies with complete voice and world-model sessions under changing speakers and scenes, concurrent learning, and resource pressure. It will report response latency, peak memory, adaptation progress, fallback latency, output quality, and verification cost, including rejected transitions and inconclusive outcomes. This prevents efficiency gains from concealing lost functionality or weakened evidence. Models and trace-driven experiments will precede live sessions on embedded devices.

My longer-term goal is to make assurance maintainable as interactive systems evolve. A developer changing a model, compiler, or scheduling policy should be able to identify the affected claims and the evidence needed to restore them. I will release runtime interfaces, verification artifacts, and traces that support this process. These interfaces would help developers judge when adaptation remains justified.

References

[1] Zexin Li, Aritra Samanta, Yufei Li, Andrea Soltoggio, Hyoseung Kim, and Cong Liu. R^3: On-device real-time deep reinforcement learning for autonomous robotics. In *IEEE Real-Time Systems Symposium (RTSS)*, 2023. doi:10.1109/RTSS59052.2023.00021.

[2] Zexin Li, Dongjoo Seo, Nikil Dutt, Jun Sheng, and Cong Liu. Real-time continual learning on embedded GPUs. In *IEEE Real-Time Systems Symposium (RTSS)*, 2026. Accepted. Conference program.

[3] Zexin Li, Weibo Zhang, Linyi Li, Arvind Easwaran, and Simin Chen. ProCerDCL: Query-efficient black-box probabilistic verification of deep learning compiler optimization. *arXiv preprint arXiv:2610.36499*, 2026. Equal contribution: Zexin Li and Weibo Zhang. Preprint.

[4] Yufei Li, Zexin Li, Wei Yang, and Cong Liu. RT-LM: Uncertainty-aware resource management for real-time inference of language models. In *IEEE Real-Time Systems Symposium (RTSS)*, 2023. doi:10.1109/RTSS59052.2023.00023.

[5] Ziliang Zhang, Zexin Li, Hyoseung Kim, and Cong Liu. BOXR: Body and head motion optimization framework for extended reality. In *IEEE Real-Time Systems Symposium (RTSS)*, 2024. doi:10.1109/RTSS62706.2024.00016.

[6] Yufei Li, Zexin Li, Yinglun Zhu, and Cong Liu. LeMix: Unified scheduling for LLM training and inference on multi-GPU systems. In *IEEE Real-Time Systems Symposium (RTSS)*, 2025. Outstanding Paper Award. doi:10.1109/RTSS66672.2025.00052.

[7] Soheil Shirvani, Aritra Samanta, Zexin Li, and Cong Liu. DuoJoule: Accurate on-device deep reinforcement learning for energy and timeliness. In *IEEE Real-Time Systems Symposium (RTSS)*, 2024. doi:10.1109/RTSS62706.2024.00019.

[8] Zexin Li, Tao Ren, Xiaoxi He, and Cong Liu. RED: A systematic real-time scheduling approach for robotic environmental dynamics. In *IEEE Real-Time Systems Symposium (RTSS)*, 2023. doi:10.1109/RTSS59052.2023.00027.

[9] Zexin Li, Jiancheng Zhang, Yufei Li, Yinglun Zhu, and Cong Liu. MixTraining: A better trade-off between compute and performance. *Transactions on Machine Learning Research*, 2026. April. Equal contribution: Zexin Li and Jiancheng Zhang. OpenReview.

[10] Zexin Li, Nikil Dutt, and Cong Liu. Orion: Enabling self-adaptive memory management for on-device online continual learning. *arXiv preprint arXiv:2605.26473*, 2026. Under review. Preprint.

[11] Daniel Casini, Tobias Blaß, Ingo Lütkebohle, and Björn B. Brandenburg. Response-time analysis of ROS 2 processing chains under reservation-based scheduling. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 6:1–6:23, 2019. doi:10.4230/LIPIcs.ECRTS.2019.6.

[12] D. Seto, B. H. Krogh, L. Sha, and A. Chutinan. Dynamic control system upgrade using the Simplex architecture. *IEEE Control Systems*, 18(4):72–80, 1998. doi:10.1109/37.710880.

[13] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004. doi:10.1007/978-3-540-30080-9_7.