MIZAN: Measure-First Adaptive Inference for Energy-Constrained Satellite Payloads

Ahmed Hedhili
SUP'COM
ahmed.hedhili@supcom.tn

Salem Essid
SUP'COM
salem.essid@supcom.tn

Ahmed Mnasri
SUP'COM
ahmed.mnasri@supcom.tn

***Abstract*—A satellite running AI must decide how much of a shifting energy, memory and time budget each inference deserves. At each frame, MIZAN adapts the model (an early exit), the processing quality (input resolution) and the inference frequency (it may skip the frame). It prices energy in closed form, corrects the menu's costs online, and lets a safety filter veto but never choose. Exiting early saves 52–57% of per-inference energy on saturated accelerators, nothing measurable at ten frames per second, where idle power dominates: run less, not shallower. On our Raspberry Pi 4B payload, the BUPT-1 CubeSat's computer, early exit runs $2.3\times$ faster than full depth for 0.9 points of accuracy, and a laptop-profiled menu underestimates latency $6.5\times$; correcting it online halved missed deadlines ($p = 0.0065$). Live on the board, the full system kept its latency bound (exceeded on 4.5–5.0% of frames against 5%) and was 4.7–5.3 points more accurate than the cheapest option, not yet significantly, for 0.7–2.1% missed deadlines against none. In a pre-registered held-out evaluation over laptop costs it caused no breach in 100 runs, though at the board's costs only with a battery reserve added afterwards. Under heat it skipped frames a lighter option could have run (39.1% of the optimum, against 92.2% for the cheapest). A step-down rule written afterwards beats every single fixed option in simulation (94.3% against 92.3%), not the best one per scenario. Onboard, adaptive inference must measure its platform, and keep measuring it.**

***Index Terms*—onboard AI inference, energy-constrained computing, early-exit neural networks, resource-aware adaptive control, orbital edge computing**

I. Introduction

Compute is moving to orbit, and with it the question of how a satellite should spend the energy it has. A payload that images continuously cannot downlink everything, so it must decide onboard what each frame is worth, under constraints that move: the battery charges and discharges with the orbit, the processor heats, and a competing task takes the deadline away.

This paper addresses *AI Inference Under Energy Constraints*, the problem statement of the Artificial Intelligence and Onboard Computing track of the IASTAM 6.0 Technical Challenge. A strategy can adapt along three levers — *which model* to run, *how often* to run it, and *how much detail* to process — and is judged on accuracy, inference time, energy consumption, memory used, and robustness to varying resources.

Adaptive inference is usually evaluated on a development machine rather than the platform that would fly it, and its energy is either not measured [1] or not separated into the board's fixed draw and the share a policy controls [2]. We build **MIZAN** (Arabic for *balance*) to measure both: an inference stage offers operating points whose costs are measured on the target, and a controller picks among them under the resources of the moment, without ever being able to override a safety limit. Our target is a Raspberry Pi 4B ($8\,$GB), the general-purpose computing payload of the BUPT-1 CubeSat [3], [4], where in-orbit measurement finds temperature and energy, not compute, to be the binding constraints [5]. Three data-centre accelerators, the hardware class orbital data-centre proposals assume [6], [7], give a contrasting regime. We make three contributions.

1) **A measurement-first methodology**: an early-exit network at two resolutions whose gated execution reproduces fixed-exit outputs exactly, a data-disjoint certification with a stated selective-error guarantee, and a provenance label on every quantity. It finds a host-profiled cost menu about $6.5\times$ optimistic on the target board, floor to floor (Sections III-D and IV-F).
2) **A veto-only controller**: a safety filter that can refuse a configuration but never choose one, an energy price computed onboard in closed form with no tuning constant, and cost feedback that corrects the menu online with a latency bound of stated miss rate (Sections III-B, IV-D and IV-G).
3) **A regime map for the three levers** (Fig. 1): on a duty-cycled payload, adapting the model buys deadline safety, not energy, and only with a cost model measured and corrected on the target. On the board, cost feedback cuts mean latency by 31% and missed deadlines by 49% for 2.7 accuracy points, though the cheapest fixed option misses fewer still (Sections IV-C and IV-G).

II. Related Work

Onboard inference. Φ-Sat-1 flew a convolutional network on a Myriad 2 vision processor to discard cloudy tiles [8]; surveys map onboard image processing [9] and orbital edge computing [10], [11]; Kodan fits every tile to the frame deadline and measures time but not energy [12]; Equinox runs a task only when its value exceeds a barrier-shaped cost of battery, temperature and queue state, across a simulated constellation [13]. We ask how a workload that is onboard should execute as its budgets move.

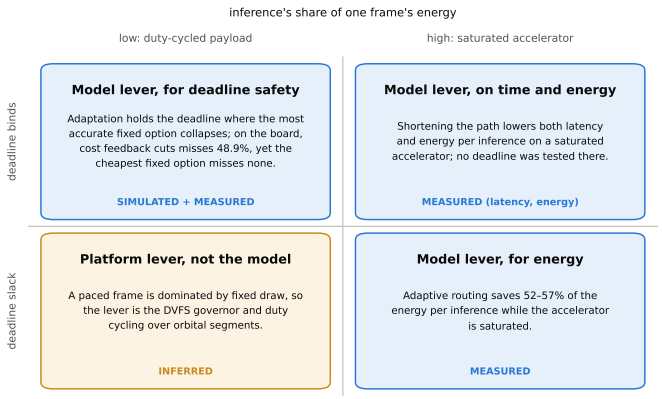


Fig. 1. The regime map: which lever pays depends on inference's share of one frame's energy and on whether the deadline binds. Badges give each cell's evidence: MEASURED on the stated platform, SIMULATED from declared assumptions, or INFERRED from a model and measurements elsewhere, not observed on the target board.

Dynamic and resource-aware inference. Early-exit networks [14], [15], [16], [17] adapt to the *input*, not to the platform, and report FLOPs or latency; a recent comparison on a Jetson Orin and a Raspberry Pi measures no energy [1], and E3C meters early exits on a Raspberry Pi 4B without separating the board's fixed draw from the share a policy controls [2]. NestDNN [18] and E4 [19] adapt to the resources available, but none bounds what its controller may do when the cost model is wrong. ReGate attributes 30–72% of neural-processing-unit energy to static power [20], energy no policy that only chooses how much to compute can see.

Risk and safety filters. Temperature scaling [21] and Learn then Test [22], [23] give calibrated and certified confidence. The Simplex architecture [24], shields [25] and predictive safety filters [26] let a layer veto a controller but never choose for it. We add where the filter's numbers come from — costs measured on the target and corrected online with an adaptive conformal bound [27] — and an energy price, the Lagrange multiplier of a budgeted sequential decision [28], [29], in closed form because the menu states the rewards, where a barrier-shaped cost [13] needs hand-set coefficients.

The gap. No prior work measures what adaptation is worth in energy on a duty-cycled flight-class board, and none closes measured costs, resource-aware control and platform energy accounting into one loop under a filter that can only veto.

III. Methodology

A. Problem Formulation

Time is divided into frames $t = 0, \dots, T-1$ of period Δ (5 s in most of our scenarios, 0.1 s in the two whose energy binds hardest). At every frame the system either skips or runs one configuration a from a menu $\mathcal{A}$ of K options profiled offline, each with accuracy $Q(a)$, mean and 95th-percentile latency $\bar{\ell}(a)$ and $\ell_{95}(a)$, resident memory $m(a)$ and energy per inference $E(a)$, estimated as $P_{\text{act}}\,\bar{\ell}(a)$ from a declared active power where it has not been measured on the target. The frozen menu has $K = 8$ options of one early-exit network, with accuracy between 69.3% and 90.7% and ℓ_{95} between 2.8 and 9.6 ms on the development host; the held-out evaluation runs its certified successor (Table I).

The controller observes $s_t = (b_t, \theta_t, m_t, d_t)$ — stored energy, processor temperature, free memory and time budget — and chooses $a_t \in \mathcal{A} \cup \{\varnothing\}$, where SKIP ($\varnothing$, $Q(\varnothing) = 0$) is a first-class action. With τ_t the controller's decision time and switching costs from measured transitions, the frame is charged

$$E_t = E(a_t) + P_{\text{act}}\,\tau_t + E_{\text{sw}}\,\mathbf{1}[a_t \neq a_{t'}], \tag{1}$$

$$\ell_t = \ell_{95}(a_t) + \tau_t + \ell_{\text{sw}}\,\mathbf{1}[a_t \neq a_{t'}]. \tag{2}$$

ℓ_t is what the filter plans with. The *realised* latency L_t is what the frame took: on the board, the measured inference time plus the measured decision time, plus ℓ_{sw} when the option changed; in simulation the menu's $\ell_{95}(a_t)$ stands in for the inference time, so $L_t = \ell_t$. A frame that runs with $L_t > d_t$ counts as a missed deadline and keeps its accuracy; skipped and vetoed frames score zero. Panels deliver $P_{\text{gen}}(t)$ outside eclipse; housekeeping P_{hk} and the value detector E_{det} draw on every frame, skipped or not:

$$\begin{aligned} b_{t+1} &= \max\{0,\ \hat{b}_t - E_t - P_{\text{hk}}\,\Delta - E_{\text{det}}\}, \\ \hat{b}_t &= \min\{B,\ b_t + \eta\,P_{\text{gen}}(t)\,\Delta\}, \end{aligned} \tag{3}$$

$$\begin{aligned} \theta_{t+1} &= \theta_\infty + (\theta_t - \theta_\infty)\,e^{-\Delta/\tau_{\text{th}}}, \\ \theta_\infty &= \theta_{\text{amb}}(t) + R\,\big((E_t + E_{\text{det}})/\Delta + P_{\text{hk}}\big). \end{aligned} \tag{4}$$

The objective is value-weighted accuracy under hard constraints:

$$\begin{aligned} \max_{a_0,\dots,a_{T-1}} \quad & \sum_{t=0}^{T-1} v_t\,Q(a_t) \\ \text{s.t.} \quad & b_{t+1} \geq b_{\min}, \quad \theta_{t+1} \leq \theta_{\max}, \\ & \ell_t \leq d_t, \quad m(a_t) \leq m_t, \quad \forall t. \end{aligned} \tag{5}$$

The value v_t is 1 when the frame images a target and 0.1 when it does not, seen only through a declared detector right on 90% of frames; results are scored with the true value, and skipped or vetoed frames count as errors. The menu carries the model and processing-quality levers, SKIP the frequency lever; SKIP does not power the payload down, since the board's fixed draw sits in P_{hk}.

B. Controller Architecture

Fig. 2 shows the loop: the controller proposes, a deterministic safety filter disposes, and the executor runs the approved option. Every layer is $O(K)$ per frame, except the conformal update (10), which re-sorts the executed option's window of W latencies, $O(W \log W)$; all of it runs unchanged on the target board.

Safety filter. A proposal is approved only if it lies in

$$\begin{aligned} \mathcal{A}_{\text{safe}}(s_t) = \{a \in \mathcal{A} :\ & \theta_t < \theta_{\max} - \delta_\theta, \quad b_t > b_{\min}, \\ & \ell_{95}(a) \leq d_t - \delta_\ell, \quad m(a) \leq m_t, \\ & E(a) + \delta_E \leq b_t - b_{\min}\}; \end{aligned} \tag{6}$$

otherwise the frame becomes SKIP. The filter can veto but never choose, so no controller, learned or hand-designed, can bypass it. The margin $\delta_\ell = 2\,\text{ms}$ covers the largest measured switching overhead (1.25 ms over 56 transitions) plus decision time.

Pareto policy. The innermost layer proposes

$$a_t = \underset{a \in \mathcal{A}_{\text{safe}}(s_t) \cup \{\varnothing\}}{\arg\max} \big[\, \hat{v}_t\, Q(a) - \lambda\, E(a) \,\big], \tag{7}$$

with $\hat{v}_t$ the detector's estimate; SKIP scores zero, so an option runs only when it is worth its energy at price λ.

Pricing energy without hindsight. λ is the Lagrange multiplier on the energy budget of the offline optimum (11). Tuning it needs foresight of the episode, but the menu states it. With H the episode length and d the declared orbit-average platform draw per frame (housekeeping plus detection, less average generation), each remaining frame may spend

$$\begin{aligned} T_t &= \frac{\max\{0,\ (b_t - b_{\min}) - (H-t)\,d\}}{H-t}, \\ a_t^\star &= \underset{\{a\,:\,E(a) \le T_t\}}{\arg\max}\ Q(a), \end{aligned} \tag{8}$$

where $a_t^\star$ is the best rung that allowance covers, or the cheapest when it covers none; when $(H-t)\,d$ exceeds the usable charge, the price drops to its minimum (Section IV-I shows the consequence at board costs). The price at which no richer rung is worth its extra energy is

$$\lambda_t = \max_{\substack{a\,:\,E(a) > E(a_t^\star),\\ Q(a) > Q(a_t^\star)}} \frac{Q(a) - Q(a_t^\star)}{E(a) - E(a_t^\star)}, \tag{9}$$

kept at its previous value when no such rung exists. Both are recomputed every frame; H and d are declared mission parameters, not tuning constants.

Hysteresis holds the current option for $D = 3$ frames, released at once if it becomes infeasible. **Cost feedback** hands every inner layer the menu at its estimated cost: from the last $W = 100$ realised latencies $L_{a,1}, \ldots, L_{a,n}$ of each executed option it replaces $\ell_{95}(a)$ in (6) by the conformal bound

$$\hat{q}_a = L_{a,(k)}, \qquad k = \big\lceil (n+1)(1-\varepsilon_t) \big\rceil, \tag{10}$$

which the next execution exceeds with probability at most $\varepsilon = 5\%$ when latencies are exchangeable. A board whose clock changes is not, so adaptive conformal inference [27] moves the level after every frame, $\varepsilon_{t+1} = \varepsilon_t + \gamma\,(\varepsilon - \mathbf{1}[L_t > \hat{q}_t])$ with $\gamma = 0.01$. An option executed fewer than 19 times keeps its menu cost.

Survival mode narrows the menu by state: SAVING to the cheaper half below 10 minutes of reserve at the unavoidable platform draw, CRITICAL to the cheapest option below 4 minutes, EMERGENCY to none; THERMAL_RECOVERY skips frames from $2\,^\circ$C below the limit until the processor is $5\,^\circ$C below it. Each layer can only narrow the choices of the layer inside it, and the filter keeps the final word.

C. Closed-Loop Evaluation Protocol

Every controller runs on a fresh copy of the same seeded environment, so differences come from decisions only, against four static references: always the most accurate option, always the cheapest, the most accurate option that fits every frame, and the best static option chosen with hindsight. The denominator is an *offline optimum*: relaxing the per-frame battery constraint to the episode's energy budget B_{ep} gives a Lagrangian that separates across frames,

$$L(\lambda) = \sum_t \max_{a \in \mathcal{A}_t} \big[v_t Q(a) - \lambda E(a) \big] + \lambda B_{\text{ep}}, \tag{11}$$

whose minimum over $\lambda \ge 0$ upper-bounds the constrained optimum by weak duality; bisection finds λ, and replaying its policy brackets the optimum. It is tight on five of seven scenarios; a second ceiling, matched to the safety filter, is used only where stated. Metrics follow the five indicators: per-frame accuracy, ℓ_{95} frame latency including controller and switching time, inference, controller and switching energy (measured, or labelled estimated), resident memory, and robustness as quality relative to the optimum while resources vary. Violations are counts, split into unavoidable ones and those caused by decisions.

D. Adaptive Inference Model and Its Menu

The task is EuroSAT RGB land-use classification [30], a standard proxy for an Earth-observation payload that labels what it images before deciding what to keep. A single MobileNetV3-Small backbone [31], initialised from ImageNet weights, carries three classifiers at increasing depth and runs at two input resolutions (48 and 64 pixels), so adaptation has three knobs that cost nothing to store: where to stop, how much input detail to use, and whether to run at all. All weights stay resident, so memory barely differs across options. Each head is temperature-scaled [21] on a split training never saw; image x leaves at the first exit k with $\max_c p_k(c \mid x) \ge \kappa_k$, and a gated run costs exactly the path it took, verified on thousands of images with zero mismatches.

Thresholds with a stated guarantee. With $\alpha = \delta = 0.10$ declared before any evaluation, candidate policies are tested in a fixed order on a certification split no earlier stage touched, by exact binomial tests of the *selective* error $\Pr(\text{wrong} \mid \text{exited early})$, stopping at the first non-rejection [22], [23]. The survivor carries the statement that, with probability at least $1-\delta$ under exchangeability with the certification region, images that leave early are wrong at most α of the time. Every option's accuracy comes from images that neither trained the model nor selected its thresholds, and its costs from running that exact checkpoint.

Two menus. Most results use the frozen eight-option menu of one training run. A pre-registered retraining over five seeds found that gating costs about 1.7 points at 64 pixels and that a geographic split costs about five; its certified successor reaches 90.15% against 91.02% at full depth (not significant) at under half the mean latency, and the held-out evaluation runs five of its rungs (Table I).

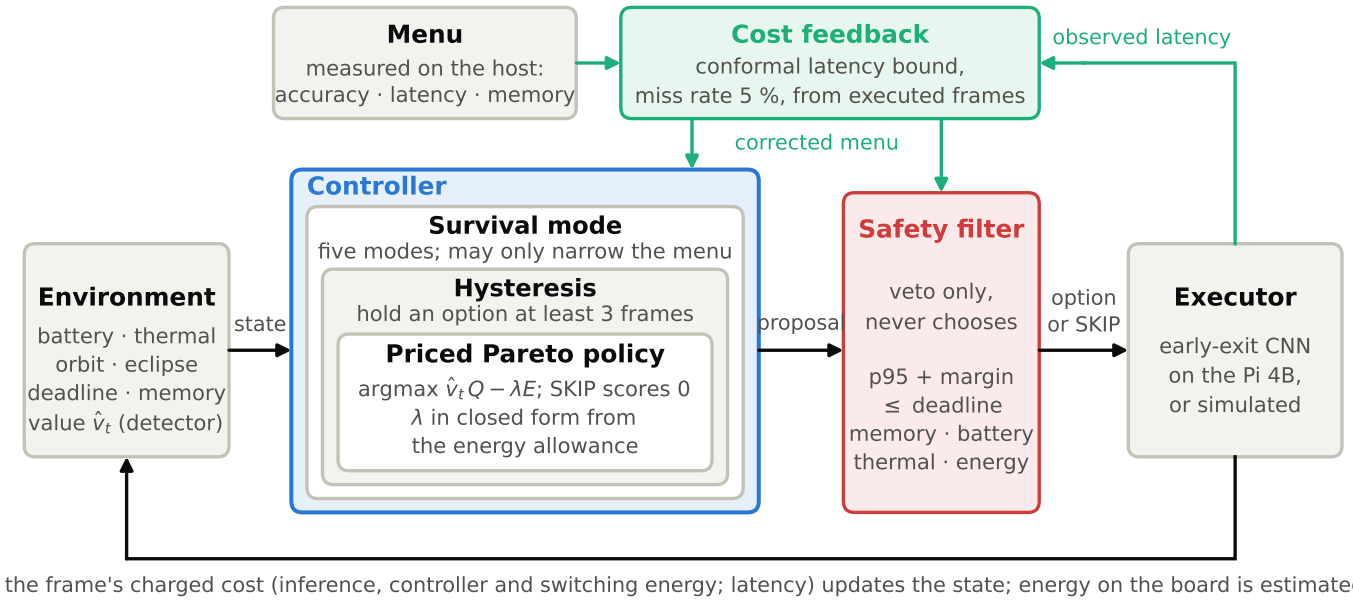


Fig. 2. MIZAN system architecture. Survival mode wraps hysteresis, which wraps the Pareto policy priced in closed form (9); together they propose an option, and the safety filter can only veto it. Cost feedback hands the controller and the filter the menu at its corrected cost (10), learned from executed frames only.

TABLE I

THE MENU THE HELD-OUT EVALUATION RUNS: FIVE RUNGS OF THE CERTIFIED SUCCESSOR. ACCURACY ON 1,625 IMAGES OF A GEOGRAPHICALLY HELD-OUT REGION; COSTS MEASURED ON THE DEVELOPMENT HOST, AS M4 RAN THEM. BOARD SESSION 7 RE-RAN ALL FIVE ON THE BOARD: THE HOST'S CLASS AND EXIT ON EVERY ONE OF THOSE IMAGES, SO EACH ACCURACY HOLDS THERE, AT 7.6–8.8× THESE MEAN LATENCIES (ℓ_{95} 20.7–64.5 MS) AND 347 MB SAMPLED PEAK MEMORY; IT MEASURED NO ENERGY.

Res.	Rung	Acc. (%)	Mean (ms)	ℓ_{95} (ms)	RSS (MB)
64	exit 1	83.88	1.55	2.08	307
64	adaptive, depth cap 2	87.63	1.86	3.38	298
48	adaptive, depth cap 2	88.31	2.09	3.72	308
64	adaptive, certified	90.15	1.91	4.25	308
64	full depth	91.02	4.58	5.51	306

IV. Experiments and Results

A. Testbed, Simulation Setup and Provenance

The **target board** is a Raspberry Pi 4 Model B (8 GB), 64-bit Linux, running inference on its ARM CPU. It ran eleven sessions, numbered 1–4 and 6–12: three with the desktop active on synthetic images, one headless on real EuroSAT images, two for energy and the M3 menu, two running the full system on real images, one measuring its energy after a reboot, and two running the live demonstration; session 5 was planned and never ran. A laptop CPU, the **development host**, profiled the menu. The **accelerators** are an NVIDIA A40 and cloud L4 and T4 instances with their own power telemetry, at batch size one. The **simulator** models a battery with a floor, first-order thermal relaxation, eclipse-gated solar generation and housekeeping charged on every frame, in seven named scenarios each built so a stated resource binds; the held-out evaluation adds *energy_trade* and two held-out scenarios. Every quantity is labelled *measured*, *simulated*, *estimated* or *declared*; board energy is measured only as a ratio, with no meter. The system changed during the work: the board sessions ran earlier versions of the controller, and only sessions 8–10 ran the system of Section III-B. Table II summarises the findings.

B. Result: What Adaptive Inference Costs on Real Accelerators

Each option runs alone on an accelerator, in randomised order, across repeated windows, either saturated or paced at a fixed frame rate. Per-inference energy is the board's power-counter difference over a window divided by completed inferences, idle draw included (Table III); subtracting an idle baseline measured inside every window isolates E_{dyn}, the energy inference itself adds.

Table III shows a large, reproducible saving on a saturated accelerator and none at 10 frames per second on the same hardware. Per frame,

$$E_{\text{frame}}(f) = \frac{P_{\text{idle}}}{f} + E_{\text{dyn}}. \tag{12}$$

E_{dyn} is at most 20.2 mJ at full depth (L4, above idle), against P_{idle}/f of 2.8–7.2 J at $f = 10\,\text{s}^{-1}$, so the model's share of a paced frame is about one percent or less; measured in the paced runs at full depth it is 1.2% on the L4 and indistinguishable from zero on the T4. The saturated saving is throughput, not lower power: the idle draw is 94.1%, 87.2% and 97.5% of a saturated full-depth inference on the A40, L4 and T4, and 86–97% of the saving is idle power charged over a shorter window. On a duty-cycled payload, adapting the model is not an energy lever; adapting the *frequency*, or powering down between bursts, is.

TABLE II

SUMMARY OF FINDINGS, EACH WITH ITS EVIDENCE STATUS; EACH ROW IS STATED IN FULL IN THE SECTION NAMED.

#	Finding	Key number	Evidence status	Section
F1	The model lever saves energy only when the accelerator is saturated	52–57% saturated; indistinguishable from 0 at 10 fps	measured (GPU boards)	IV-B
F2	A host-profiled menu is optimistic on the board	about 6.5× at the floor	measured (Pi 4B)	IV-F
F3	Cost feedback recovers deadlines on the board	−48.9% misses ($p = 0.0065$), −31.1% mean latency ($p = 0.0011$)	measured (Pi 4B)	IV-G
F4	The cheapest fixed option still misses fewer deadlines	0 vs 23 misses; 0 vs 10 and 30 of 1,440 with the full system	measured (Pi 4B)	IV-G
F5	The onboard energy price needs no hindsight where energy binds	98.4% of optimum; hindsight price and cheapest option 97.6%	simulated	IV-D
F6	Inference is a few percent of a paced frame's energy on the board	0.2–3.4%	estimated	IV-F
F7	The onboard price removes the breaches a handset price causes	454 (policy) and 154 (full stack) to 0	simulated	IV-H
F8	The board makes the host's decision on every image	2,560 of 2,560 paced frames; all 1,625 test images, all ten classes	measured (Pi 4B)	IV-F
F9	On the board, energy per inference follows its duration	full depth 2.6–3.1× an early exit; switching off pays after about 40 s	measured, as a ratio (Pi 4B)	IV-F
F10	Held out: no breach, but quality lost under heat	0 breaches in 100 runs; heat: 39.1% vs. 92.2% (cheapest)	simulated over host costs, preregistered	IV-I
F11	Stepping down beats any single fixed option, not the best one per scenario	94.3% on fresh seeds vs. 92.3% for the best single fixed option (the cap-2 rung on every frame); no breach	simulated, exploratory	IV-I
F12	Early exit cuts inference time on the board	certified rung 2.3× faster than full depth on the mean (16.9 vs. 39.5 ms), 3.3× on the median, for 0.9 points of accuracy	measured (Pi 4B)	IV-F
F13	Live on the board, the full system holds its latency bound and gains accuracy, not yet significantly	bound exceeded on 4.54 and 5.0% (budget 5%); 88.1 and 88.6% vs. 83.3% for the cheapest (McNemar $p = 0.080$, 0.058)	measured (Pi 4B), two sessions	IV-G

TABLE III

MEASURED TOTAL ACCELERATOR ENERGY PER INFERENCE AT 64 PIXELS, IDLE DRAW INCLUDED, ADAPTIVE ROUTING AGAINST FULL DEPTH, WITH 95% CONFIDENCE INTERVALS OVER THREE TRIALS. SATURATED: THE ACCELERATOR IS THE BOTTLENECK; PACED: 10 FRAMES PER SECOND, BOARD POWERED BETWEEN THEM.

	A40	L4	T4
Idle board power (W)	71.9	28.6	28.0
Full depth, sat. (mJ)	324.4	157.2	157.8
Adaptive, sat. (mJ)	151.5	67.7	75.8
Saving, sat. (%)	53.3	56.9	52.0
95% interval	[50.4, 56.2]	[56.2, 57.6]	[42.3, 61.6]
Saving at 10 FPS (%)	–	0.65	−0.05
95% interval	–	[−2.1, 3.4]	[−0.2, 0.1]

C. Result: When Adapting Pays on Energy

What adaptation can save is bounded by inference's share of the energy the platform spends anyway. We sweep that share by varying housekeeping power on one scenario and score each controller as a share of the offline optimum (Fig. 3a). Where the energy budget binds, a price tuned with hindsight reaches 96.7–99.0%, against 96.7–97.6% for the best fixed configuration with the same hindsight: adaptation is at best a couple of points ahead. The price computed onboard matches or beats the tuned one at the two looser binding points (99.3% and 98.4%) and falls short at the tightest (91.7% against 96.7%), where every controller, fixed ones included, spends through the battery floor (rings). The accelerators say the same from the hardware side (Fig. 3b). The case for adapting onboard therefore rests on deadlines under changing resources, not on energy.

D. Result: The Energy Price Need Not Be Tuned

Where energy binds, an untuned default price reaches only 86.7–93.5% of the offline optimum, and the price that does better needs hindsight. On the one scenario of our seven where energy binds, the same decision policy ran with three price sources. The onboard price of (8)–(9) matches the hindsight price (97.6%, every frame executed, no breach) where the untuned default (87.2%) loses hundreds of frames to the filter and causes 454 decision-caused breaches, and under the current rule it reaches 98.4%, above hindsight, because it can move during the episode. Inside the *full* stack the interim rule reaches 93.2%, 4.4 points below the same rule alone; under the current rule, on the M3 menu and M4's held-out seeds, the gap is 6.5 points (100.0 against 93.5%, Table IV): survival mode and hysteresis buy constraint safety the price already bought. In simulation the declared draw d is the scenario's true orbit-average draw, so M4 never tested a wrong one. Misdeclared (E3, exploratory), planning with less energy than there is costs quality, 8.1 and 14.9 points on *compute_bound* with d 20 and 50% too high and 2.4 and 8.6 on *energy_trade* with its surplus as much too low; planning with more costs nothing measurable, but at half the true drain the price alone caused 3 battery breaches on *compute_bound*, which the full stack prevented. The result shows the rule finds the right rung without hindsight, not that adapting pays: running the cheapest option unconditionally scores within a fraction of a point. On *energy_trade*, built so the optimum moves between three rungs, always running the certified rung reaches 99.4% and the full system 98.3% (99.3% with a perfect value detector):

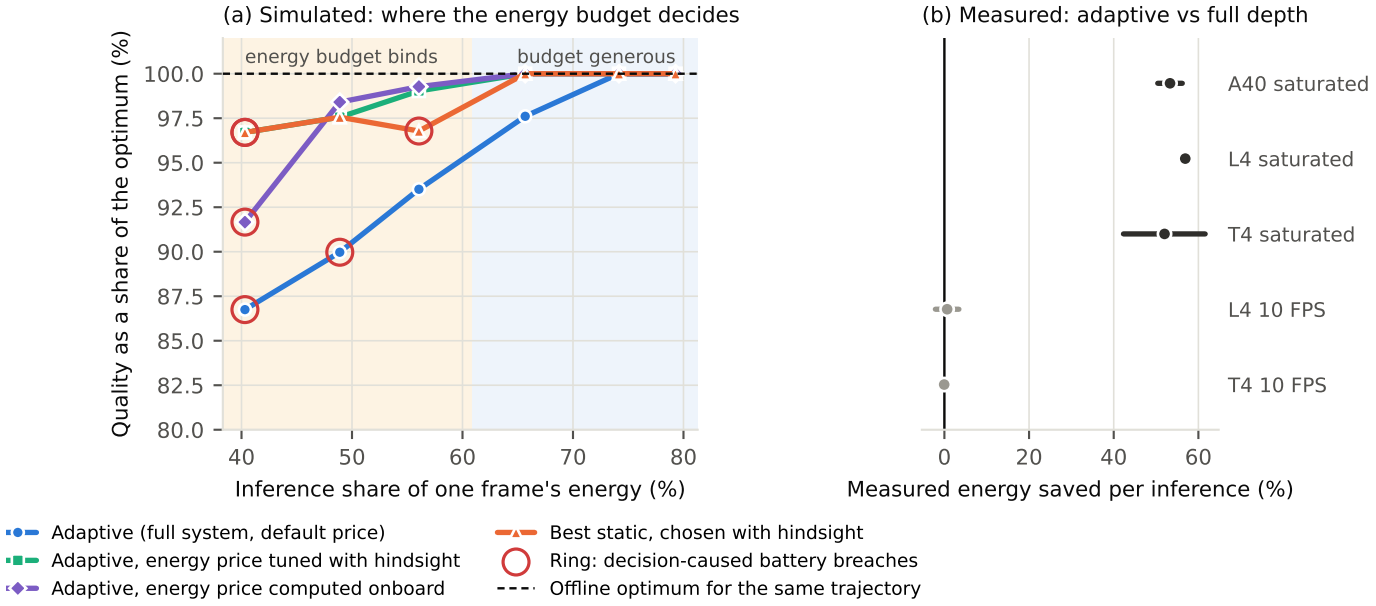


Fig. 3. When adapting pays on energy. (a) Simulated on *compute_bound* with the frozen menu: quality as a share of the offline optimum against inference's share of a frame's energy (axis from 80%); a ring marks decision-caused battery breaches. Below 40% housekeeping alone breaches the floor; the board's estimated share (0.2–3.4%, Section IV-F) lies there. (b) Measured on three accelerators: total energy saved per inference by adaptive routing, 95% intervals; the A40 was not run paced.

on a menu whose certified rung is within 0.9 points of full depth at 42% of its energy, adapting is worth under a point.

E. Result: Robustness to a Varying Time Budget and to a Wrong Menu

A varying time budget. With energy and memory non-binding, seeded bursts tighten the deadline over part of a 600-frame run. The adaptive controller matches the per-frame best score in every setting; the most accurate fixed option drops from 90.7% to 63.5% accuracy at 30% of frames under pressure, and to 0% once pressure is constant, because the filter refuses it; the best fixed option chosen with hindsight survives only by always running the fastest option.

A wrong menu. When execution is slower than the menu says, the filter approves work that cannot finish in time. At a 1.5× slowdown a menu-blind controller misses all 200 contended deadlines, so its on-time accuracy is 60.4%; with cost feedback it misses 2 and keeps 86.4% on time. At the 8–9× of the contended board no option fits and the only safe action is to skip: cost feedback buys *deadline safety, not quality*.

F. Result: The Target Board — Transfer Error and Platform Behaviour

The menu is optimistic by about 6.5×, floor against floor. The cheapest option, exit 1 at 64 pixels, runs on the board at a first quartile of 9.50 ms in one session and 9.41 ms twenty hours later — a floor reproducible to 0.9% — against 1.45 ms on the re-measured host: 6.56× and 6.50×. The median moved 56% between them on ambient load alone, so a menu entry is tested against uncontended frames on both machines. A ratio this far beyond the 1.5× that already defeats a menu-blind controller makes cost feedback a precondition for the margins of (6) to mean anything.

Session 4: headless, real images. The floor ratio is 6.60×. The slow tail came from the CPU governor: for twenty minutes the clock sat at 600–900 MHz and every arm ran about 1.9× slower. Deadline misses followed the clock, not the controller: 15 and 12 for the two adaptive arms in the slow block, none at full clock, and none for the cheapest option. The governor was left at `ondemand`, a gap in the design: the controller does not observe the clock.

Session 7: the M3 menu on the board. The board takes the host's class and exit on all 2,560 paced frames of session 4 and, in session 7, on all 1,625 test images for 10 options: the host's accuracy carries over on all ten classes. The five M3 rungs run at 7.6–8.8× the host's mean latency: 12–17 ms early, 39.5 ms at full depth.

Decision cost, heat and the energy share. Run means are 0.02–0.03 ms per decision for a fixed option, 0.08–0.13 ms for the adaptive policy and 0.35–0.55 ms with cost feedback, whose 1.19 ms maximum plus a 1.25 ms switch crossed the declared 2 ms margin δ_ℓ on 30 of 360 frames at the low clock. A margin measured on the board would need 2.25 ms to cover 99% of those frames (maximum 2.44 ms; 1.91 and 2.05 ms at full clock). No sample throttled against an 80 °C limit (hottest 68.1 °C without a fan, 75.0 °C saturated with one). One inference is 0.2–3.4% of a paced frame's energy, *estimated* from a published base draw [32] and our declared active power.

Session 6: board energy, as a ratio. With no meter, an oscilloscope read the sag of the board's own 5 V rail through

its supply cable. Full depth costs 2.6–3.1× the energy above idle of an early-exit rung; the four early rungs cannot be told apart, and each sags the rail by about 50 mV, so energy follows duration, as on the accelerators. Switching off pays only for gaps over about 40 s. Joules would need a calibration with a known load.

G. Result: Cost Feedback on the Target Board

A second board session drives the controller with the board's own transfer error, cost feedback disabled and enabled in interleaved runs. Feedback cuts mean latency by 31.1% and deadline misses by 48.9%, for 2.7 accuracy points; every trial with feedback beats every trial without it on mean latency ($p = 0.0011$, exact permutation test, six trials per arm), and misses fall from 7.5 to 3.8 per trial ($p = 0.0065$). Without it the controller stayed on one option budgeted at 6 ms and executed at 26–27 ms; with it, it spread across seven. On real images in session 4 it helped only while the clock was low: 13.8% lower mean latency and misses from 15 to 12, for 2.22 measured accuracy points ($p = 0.008$, McNemar).

From an average to a bound. That rule learned an exponential average of observed over predicted latency, which settles near the mean and understates the tail: replaying session 4 in time order, its bound was exceeded on 33.2% and 36.5% of frames. The adaptive conformal bound (10), which the controller now runs, is exceeded on 5.0% and 5.1% against its 5% budget, across the clock change. A replay tests the predictor, not the closed loop: no frame is re-decided, and every board result above is the averaging rule's.

Against the cheapest fixed option. Always running the cheapest option missed no deadline in either session. What the corrected controller buys over it, about 1.5 accuracy points, costs 23 missed deadlines. The cause is the menu: with a narrow accuracy spread, adaptation has little to win and a latency tail to risk.

The full system live on the board. Sessions 8 and 9 ran M4 as frozen, with the board's menu costs and the conformal bound, against the cheapest option on *stable* with real test images, 1,440 frames per arm; the second, decided after the first, added E1. Predictions preceded both; both ran after our own cut-off for board results. The bound held live, exceeded on 4.54% and 5.0% of the frames it covered. M4 was more accurate, 88.1% in session 8 and 88.6% in session 9 against 83.3%, but on 240 images not significantly (McNemar $p = 0.080$ and 0.058), and it missed 10 and 30 deadlines against none, nearly all in an option's first 19 runs, before it had a bound: the certified rung ran at its menu cost, overran, and its first bound excluded it. Decisions took 0.15–0.17 ms on mean; E1 tracked M4 closely (88.3% against 88.6%, 29 against 30 misses). The board slowed through the evening (cheapest option 12.9, then 16.2 ms); rebooted, it ran at 9.9 ms, and session 10 (120 images) saw M4 keep the certified rung on 170 of 180 frames without a miss, at 1.45× the cheapest option's rail-sag energy per inference, speed and images confounded.

The live demonstration. Sessions 11 and 12 ran it on the board with real images and load, battery and heat scripted, averaging cost feedback and the step-down rule: six runs kept 91.6% accuracy, with 33 misses in 360 frames; a forced kill lost no recorded frame.

H. Ablation

What each layer buys. Removing one layer of Survival(Hysteresis(Pareto)) at a time, filter always on, changes nothing where no resource binds. Where one binds, survival mode buys constraint satisfaction, not quality: the thermally blind variant scores 84.7% against 64.9% on *thermal_ramp*, and the onboard price takes *compute_bound*'s decision-caused battery breaches from 454 (policy alone) and 154 (full stack) to 0. Section IV-I turns that finding into a rule (E1).

I. Held-Out Evaluation

Protocol. The system of Section III-B, with the M3 menu, the current price rule and conformal cost feedback, was frozen and its tests were written down in the repository before it ran, once, in simulation over host costs: the eight development scenarios on ten held-out seeds, and two scenarios never run before, *payload heat* and *contact pass*. Four tests were fixed in advance: P1, no decision-caused battery, thermal or memory breach; P2, the onboard price within two points of the hindsight price; P3, quality at least that of the cheapest fixed option; and T1, adopting the thermal variant if it does better on the held-out thermal scenario without more breaches.

Outcome (Table IV). **P1 holds**: no decision-caused breach in any of the 100 runs. **P2 holds** as read one-sided before the run: the onboard price is never more than 0.013 points below the hindsight price, and is 18.3 and 4.6 points above it where energy binds. **P3 fails under heat**: the system falls below the cheapest option on *thermal_ramp* (69.0 against 70.7%) and far below it on payload heat (39.1 against 92.2%), where thermal recovery skipped 60% of the frames instead of stepping down to a cooler option. **T1 adopts the variant**: 92.8 against 39.1% on payload heat, with no breach for either, although on *thermal_ramp* the variant crosses the limit once per seed where the original never does. Where deadlines or memory move, the system holds 99.5–99.6% of the optimum, against 96.3–96.4% for the best fixed option chosen with hindsight.

Why P3 failed. The thermal condition of (6) does not depend on the option, unlike the battery condition, so near the limit the filter can only approve every option or veto every one. Approving a only if its predicted temperature $\hat{\theta}_{t+1}(a)$ from (4) stays below $\theta_{\max} - \delta_\theta$ (E5, exploratory) runs every frame M4 skipped on payload heat, for 98.8% of the optimum with no breach, and leaves the other scenarios unchanged. A one-frame check cannot see a crossing two frames ahead, so *thermal_ramp* keeps the variant's 10 breaches; and since the simulator uses the same thermal model as the check, this is an upper bound.

At the board's costs (E2, exploratory). With the menu at the costs measured on the board, each inference costs about eight times more. Where deadlines or memory bind, no rung fits; where energy binds, **the system no longer keeps its**

TABLE IV

M4, the held-out evaluation, run once with the system frozen: share of the offline optimum (%), mean over ten held-out seeds. *Variant*: the pre-declared thermal variant. *Onboard* and *Hindsight price*: the Pareto policy alone, without survival mode, hysteresis or cost feedback, priced by (8)–(9), or at the multiplier the offline optimum selects for that same run. *Best fixed*: per scenario, the fixed option with the highest mean, chosen with hindsight. *E1*: written after the run, exploratory (Section IV-I). Bold: below the cheapest option. † Decision-caused breaches. Simulated over host costs.

Scenario	M4	Variant	Onboard price	Hindsight price	Best fixed	Cheapest	E1
stable	100.0	100.0	100.0	100.0	100.0	92.2	100.0
orbital	100.0	100.0	100.0	100.0	100.0	92.2	100.0
draining	70.1	70.1	74.5	74.5	74.5	68.6	74.5
thermal ramp	**69.0**	76.0†	76.6†	76.6†	76.6†	70.7	75.0
sudden drop	99.6	99.6	99.6	99.6	96.4	92.3	99.6
deadline pressure	99.5	99.5	99.5	99.5	96.4	92.3	99.5
compute bound	93.5	93.5	100.0	81.7†	99.9	93.0	100.0
energy trade	98.2	98.2	98.7	94.1	99.4	92.4	98.3
payload heat (held out)	**39.1**	92.8	71.8†	71.8†	99.1	92.2	92.8
contact pass (held out)	99.6	99.6	99.6	99.6	96.3	92.2	99.6
Mean, ten scenarios	86.9	92.9†	92.0†	89.7†	93.9†	87.8	93.9

constraints: on *compute_bound* it breaches the battery floor on 10,308 frames, because the price has no rung below the cheapest and, once the reserve is spent, treats energy as free. P1 therefore holds only over host costs. An **energy reserve in the filter** (E8, exploratory) approves an option only if the charge it leaves lets the idle payload keep the floor: the 10,308 breaches become none, *compute_bound* runs 16.4% of its frames, and every other scenario is unchanged. Under payload heat the board-cost system falls to 27.8% against the cheapest option's 92.2%.

A rule written afterwards (E1, exploratory). Averaged over the ten scenarios, the frozen system (86.9%) is below the best single fixed option, the 64-pixel cap-2 rung (91.7%), and the cheapest (87.8%). E1 steps down instead of skipping, drops SAVING and CRITICAL, and counts the controller's own draw in the price. On M4's data it reaches 93.9%, level with the best fixed option chosen per scenario with hindsight but without that option's breaches; on ten fresh seeds it reaches 94.3%, 2.01 points above the best single fixed option (95% interval $[1.93, 2.08]$) and 0.14 points below the best one per scenario ($[0.07, 0.22]$). It was designed after payload heat failed, so these are new trajectories, not held-out scenarios. Adding each change alone shows that dropping SAVING and CRITICAL gives all of its gain where energy binds: the price and the survival levels reacted to the same falling battery, and scarcity should reach decisions through one channel. That channel must also be able to skip, which the price cannot yet: at the board's costs, on *energy_trade*, E1 falls to 48.0% where M4 keeps 80.7%.

Against the five indicators (Table V), counting skipped frames as wrong, M4 is the least accurate of the four policies (77.5% per frame), because of payload heat, and E1 the most (85.1%). MIZAN does not minimise inference time or energy: it spends both where the deadline and the battery allow. Its 5.53 ms is the full-depth rung's, run where the deadline admits it, and no deadline is missed in the hundred runs.

Deviations, each fixed before the run: M4 ran on host costs, 7.6–8.8× optimistic on the board, because the board profile arrived after the menu was frozen; the price test is read one-sided, since two-sided it fails where the onboard price is better; and the power-state test was dropped.

TABLE V

MIZAN against the five indicators of Problem 2 on the hundred held-out runs: each scenario's mean over its seeds, then the mean over the ten scenarios unless stated. *Best single*: the 64-pixel cap-2 rung on every frame. E1 is exploratory. Simulated over host costs, except the rows marked board (measured). Energy is *estimated*, as Problem 2 evaluates it: the declared 2 W over host latency.

Indicator	Cheapest	Best single	M4	E1
Accuracy per frame, skips wrong (%)	79.8	83.4	77.5	85.1
board, per inference (%)	83.3[a]	87.6	88.1[a]	88.3[a]
Inference time, ℓ_{95} frame latency, worst scenario (ms)	2.10	3.40	5.53	5.53
board, ℓ_{95} per inference (ms)	20.7	29.9	36.0[a]	37.4[a]
Estimated energy per executed frame (mJ)	3.15	3.77	7.68	7.28
Memory, sampled peak RSS (MB)	306.9	298.0	307.5	307.6
board (MB)	347.7	347.0	—[b]	—[b]
Robustness: share of optimum, mean (%)	87.8	91.7	86.9	93.9
minimum over scenarios (%)	68.6	71.7	39.1	74.5
decision-caused breaches	0	0	0	0

[a] Measured live on 240 test images: the cheapest option and M4 in session 8, E1 in session 9 (where the cheapest option also scored 83.3%); the other board cells profile each rung on all 1,625. [b] Every rung measured 347.0–347.7 MB on the board, all weights resident.

V. Conclusion

Adapting the model saves about half the energy per inference on a saturated accelerator and nothing measurable at the frame rates a satellite payload uses, because the platform's fixed draw dominates each paced frame: on a duty-cycled payload the energy lever is how often the payload is powered, not how deep the network runs. What adapting buys is deadline safety, and only with a cost model measured on the machine that runs it: on our Raspberry Pi payload a host-profiled menu was 6.5× optimistic, and correcting it online recovered a third of the latency and half the missed deadlines. On the board,

energy follows duration (full depth 2.6–3.1× an early exit) and switching off pays only for gaps over about 40 s. On scenarios it had never seen, in simulation over host costs, the controller kept every constraint and priced energy without hindsight, but its thermal check cannot tell a cool option from a hot one, so thermal recovery gave up quality the cheapest option keeps; checking each option's predicted temperature repairs that in simulation, though not a crossing two frames ahead. A rule written afterwards that steps down instead beats every single fixed option on fresh seeds of the same scenarios, but not the best one per scenario. Its gain over the adopted variant comes from one change, letting the price alone answer scarcity: scarcity should reach decisions through one channel, provided that channel can choose to skip, which at the board's costs the price cannot yet; there, a reserve in the safety filter keeps the battery floor. Live on the board, the full system held its latency bound and gained accuracy for a few missed deadlines; the reserve and the step-down rule have not been tested there where they would act, and each needs held-out scenarios of its own.

Next steps. First, seed each option's bound with earlier board samples, run the board where a resource binds, and calibrate the rail sag into joules. Second, test the step-down rule and the predictive thermal check on new held-out scenarios. Third, make the payload's power state an action: SKIP keeps the board on and saves only the inference's small share, whereas powering down saves the platform's whole draw, worth it for gaps over the 40 s a boot costs, as in eclipse.

References

[1] N. Fernandez, I. Valdes, S. Van Vaerenbergh, I. de la Iglesia, and J. Arratibel, "A comparative study of CNN optimization methods for edge AI: Exploring the role of early exits," *arXiv preprint arXiv:2604.14789*, 2026.

[2] Y. Zhao, S. Li, and X. Feng, "Lightweight remote sensing scene classification on edge devices via knowledge distillation and early-exit," in *Proceedings of the 33rd ACM International Conference on Multimedia (MM)*, 2025.

[3] Tiansuan Constellation, "Tiansuan experiment platform: Specifications of the BUPT-1 satellite," GitHub repository TiansuanConstellation/TiansuanExperimentPlatform, [Online]. Available: https://github.com/TiansuanConstellation/TiansuanExperimentPlatform. Accessed: Sep. 21, 2026.

[4] C. Wang, Y. Zhang, Q. Li, A. Zhou, and S. Wang, "Satellite computing: A case study of cloud-native satellites," *arXiv preprint arXiv:2307.08530*, 2023.

[5] R. Xing, M. Xu, A. Zhou, Q. Li, Y. Zhang, F. Qian, and S. Wang, "Deciphering the enigma of satellite computing with COTS devices: Measurement and analysis," in *Proc. ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2024.

[6] B. Agüera y Arcas, T. Beals, M. Biggs, J. V. Bloom, T. Fischbacher, K. Gromov, U. Köster, R. Pravahan, and J. Manyika, "Towards a future space-based, highly scalable AI infrastructure system design," *arXiv preprint arXiv:2511.19468*, 2025.

[7] R. Thummala and G. Falco, "When to compute in space," *arXiv preprint arXiv:2512.17054*, 2025.

[8] G. Giuffrida, L. Fanucci, G. Meoni, M. Batic, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Vercruyssen, G. Furano, M. Pastena, and J. Aschbacher, "The Φ-Sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation," *IEEE Transactions on Geoscience and Remote Sensing*, 2022.

[9] A. Duggan, B. Andrade, and H. Afli, "Advancing earth observation: A survey on AI-powered image processing in satellites," *arXiv preprint arXiv:2501.12030*, 2025.

[10] C. Wu, Y. Li, M. Xu, C. Guo, Z. Yin, W. Gao, and C. Chi, "A comprehensive survey on orbital edge computing: Systems, applications, and algorithms," *arXiv preprint arXiv:2306.00275*, 2023.

[11] B. Denby and B. Lucia, "Orbital edge computing: Nanosatellite constellations as a new class of computer system," in *Proc. 25th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[12] B. Denby, K. Chintalapudi, R. Chandra, B. Lucia, and S. Noghabi, "Kodan: Addressing the computational bottleneck in space," in *Proc. 28th ACM Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), Vol. 3*, 2023, pp. 392–403.

[13] A. K. Erol and D. Mahajan, "Equinox: Decentralized scheduling for hardware-aware orbital intelligence," *arXiv preprint arXiv:2604.19958*, 2026.

[14] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. International Conference on Pattern Recognition (ICPR)*, 2016.

[15] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *Proc. International Conference on Learning Representations (ICLR)*, 2018.

[16] Y. Kaya, S. Hong, and T. Dumitras, "Shallow-deep networks: Understanding and mitigating network overthinking," in *Proc. International Conference on Machine Learning (ICML)*, 2019.

[17] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, 2022.

[18] B. Fang, X. Zeng, and M. Zhang, "NestDNN: Resource-aware multi-tenant on-device deep learning for continuous mobile vision," in *Proc. 24th Annual Int. Conf. Mobile Computing and Networking (MobiCom)*, 2018.

[19] Z. Zhang, Y. Zhao, M.-C. Chang, C. Lin, and J. Liu, "E4: Energy-efficient DNN inference for edge video analytics via early-exit and DVFS," *arXiv preprint arXiv:2503.04865*, 2025.

[20] Y. Xue and J. Huang, "ReGate: Enabling power gating in neural processing units," in *Proc. 58th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2025.

[21] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. International Conference on Machine Learning (ICML)*, 2017.

[22] A. N. Angelopoulos, S. Bates, E. J. Candès, M. I. Jordan, and L. Lei, "Learn then test: Calibrating predictive algorithms to achieve risk control," *arXiv preprint arXiv:2110.01052*, 2021.

[23] Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[24] L. Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[25] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. AAAI Conference on Artificial Intelligence*, 2018.

[26] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.

[27] I. Gibbs and E. Candès, "Adaptive conformal inference under distribution shift," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[28] A. Badanidiyuru, R. Kleinberg, and A. Slivkins, "Bandits with knapsacks," *Journal of the ACM*, vol. 65, no. 3, pp. 13:1–13:55, 2018.

[29] L. Huang and M. J. Neely, "Utility optimal scheduling in energy-harvesting networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 4, pp. 1117–1130, 2013.

[30] P. Helber, B. Bischke, A. Dengel, and D. Borth, "EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 7, pp. 2217–2226, 2019.

[31] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[32] T. Pfandzelter, N. Bauer, A. Leis, C. Perdrizet, F. Trautwein, T. Schirmer, O. Abboud, and D. Bermbach, "Trabant: A serverless architecture for multi-tenant orbital edge computing," *arXiv preprint arXiv:2504.08337*, 2025.