Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Adaptive Edge Intelligence for Smart Traffic Monitoring: Runtime Detector Selection from Device Resources, Data Characteristics and Application SLAs

Pooja G B[1] and Dr Neelambike S[2]
Department of M.Tech Data Engineering, GM University, Davanagere, India

Abstract **- Edge devices that run deep-learning inference for smart traffic monitoring face two kinds of variation at once: the computing resources available on the device change over time, and the difficulty of the incoming video changes with illumination, blur and scene density. This paper describes an adaptive edge-intelligence framework that selects, at run time, one detector from a pool of three ONNX models (YOLOv8n INT8, YOLOv8s FP16 and RT-DETR FP32). The decision combines a device-resource vector, a data-characteristic vector computed from the frames (blur, luminance, noise, entropy and edge density) and application service-level agreements (SLAs). Models that violate hard constraints are pruned, the rest are ranked by TOPSIS with weights that adapt to the observed conditions, and a hysteresis controller (dwell time, cooldown and score margin) limits switching. The framework was implemented as a FastAPI service with an edge agent and a web dashboard and evaluated, as a preliminary feasibility study, on a 32-frame scripted scenario (four phases of image and device conditions) on a CPU-only host, with model-based accuracy estimates. Against a resource-only policy, the full framework had 32.9% lower mean latency and made no model switch instead of five, but it was identical to the fixed balanced model because the SLA constraints left that model as the only viable choice on every frame, so the benchmark does not show a benefit over the best fixed model. In an ablation run under a looser SLA and a different decision configuration, mean latency fell from 73.4 ms with device information alone to 44.7 ms with all sources enabled. The scenario is short and scripted, accuracy is modelled and not measured, and the hardware is simulated, so the results describe the decision logic and not real-world detection performance.**

Index Terms - Edge intelligence, adaptive model selection, TOPSIS, multi-criteria decision making, object detection, smart traffic monitoring, ONNX Runtime.

I. INTRODUCTION

Inference is moving from data centres to the devices that produce the data. Roadside cameras, gateways and embedded boards now run neural networks locally to meet latency limits and to work without a continuous network link [1][2]. Such devices are heterogeneous and resource-constrained, and their state changes over time: CPU load, free memory, temperature and battery all vary. Compression techniques such as integer quantisation [3] reduce the cost of a model, but one optimised model used for the whole period of operation remains a compromise.

Traffic video adds a second source of variation. Daylight, dusk, rain-induced blur, sensor noise and vehicle density change how hard detection is from one moment to the next. A lightweight model can lose accuracy on degraded frames, whereas a heavy model wastes computation and may miss its frame deadline on easy ones. A policy that looks only at device resources can run a heavy model on an empty road because the processor is idle, and can fail to upgrade in rain because the processor is not saturated. A policy that switches models too eagerly also pays a loading cost and may oscillate.

This paper addresses model-level adaptation that is driven jointly by device resources, data characteristics and application requirements, with explicit protection against unstable switching. The contributions are:

1) a decision pipeline that combines hard-constraint pruning, condition-dependent weights and TOPSIS ranking over a three-model pool;
2) an anti-flapping controller with dwell time, cooldown and a memory-dependent switching margin;
3) a working implementation (service, edge agent, dashboard, benchmark and ablation harness); and
4) a preliminary six-strategy benchmark and a five-experiment ablation, reported with their limits, including the result that the full framework did not outperform the best fixed model.

II. RELATED WORK

Early systems for adaptive mobile vision choose among model variants under resource limits. MCDNN trades accuracy against resource use for continuous video [4]; NestDNN builds multi-capacity models that are selected from the resources available [5]; ApproxDet adapts object detection to both content and resource contention [6]; and Chameleon adapts the configuration of video analytics [7]. These systems establish that content and resource information both affect the best operating point. Their selection logic, however, is usually a single scoring or profiling rule, and the stability of the switching decision is seldom the subject of the evaluation.

More recent systems adapt models for edge video analytics in several ways. Ekya shares an edge server's GPU between inference and continuous retraining of the deployed

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

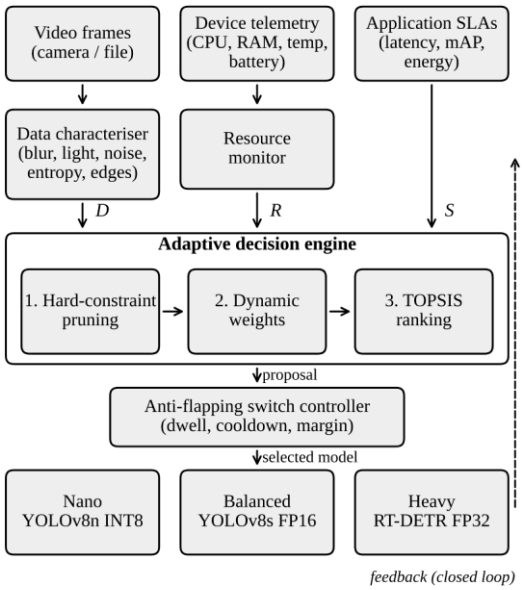
models [8], and RECL keeps a zoo of expert models, picks the one that suits each camera's current scene with a learned gating network, and retrains when none fits [9]. AdaSpring compresses a network at run time to suit the current device context [10]. CASVA chooses the video encoding configuration (frame rate, quantisation and resolution) of each segment to suit network bandwidth and content [11]. RAVAS selects among networks of different accuracy and cost with Q-learning, based on the load of a shared edge GPU, and allocates resources to limit interference between streams [12]. AxiomVision selects a visual model with an accuracy guarantee for perspective-aware video analytics [13], and EdgeMA adapts models for real-time video analytics on edge devices [14]. Other recent work adapts detection frameworks for edge deployment
[15] or partitions a model across device and server [16]. This list is not exhaustive.

The present work differs from these systems in three respects. First, it needs no online retraining and no GPU: it chooses among three pre-built ONNX models of different precision, and describes the scene with inexpensive, interpretable image measures (blur, noise, entropy, edge density, luminance) and not with a learned gating network or a load signal alone. Second, the decision also uses device temperature, battery and application SLAs, and removes models that cannot meet accuracy, memory and latency constraints before ranking. Third, switching stability is a designed and logged part of the controller (dwell, cooldown and margin). No numerical comparison with these systems was made, because they target GPU, multi-stream or network-streaming settings and were not re-implemented here; the claim of difference is one of design and not of measured performance.

III. PROPOSED FRAMEWORK

A. Overview

Three inputs are combined at each decision: a resource vector **R** (CPU, memory, temperature, battery), a data vector **D** computed from the current frame, and an SLA vector **S** (latency, minimum mAP, energy). The pool M contains three ONNX detectors [17]: Nano (YOLOv8n, INT8) [18], Balanced (YOLOv8s, FP16) and Heavy (RT-DETR, FP32) [19]. Fig. 1 shows the data flow. The decision engine proposes a model, the switch controller approves or suppresses the proposal, and measured performance feeds back into the next decision.



Condition	Threshold	Weight increase
Low light	luminance < 40	Robustness +0.10
CPU saturation	CPU > 80%	Latency +0.20, resource +0.15
Thermal	SoC > 75 °C	Resource +0.25
Low battery	battery < 20%	Resource +0.35

Fig. 1. Architecture of the adaptive decision pipeline.

B. Problem Formulation

At decision time t the system observes the resource vector R_t, the data vector D_t and the SLA vector S. It must choose a model m_t from the pool M so that the accuracy-latency-resource trade-off is as good as possible, subject to the hard constraints of Section III-D and to the stability rule of Section III-F, which limits how often m_t may differ from m_{t-1}. The problem is therefore a constrained multi-criteria selection that is repeated over time, and not a one-off optimisation.

C. Data Characterisation

Frame quality and complexity are measured with inexpensive classical operators. Blur is the variance of the Laplacian [20], noise is estimated with the Immerkær filter [21], and Shannon entropy [22] is computed over a 64-bin grey-level histogram. Edge density ρ is obtained from Canny edges [23] with an Otsu-derived threshold [24]. Scene complexity combines entropy and edge density:

$$B = \mathrm{Var}(\nabla^2 I) \quad (1)$$

$$H = -\sum_{k=1\ldots64} p_k \log_2 p_k \quad (2)$$

$$C = 0.6\cdot(H / 6.0) + 0.4\cdot\min(1, \rho/0.15) \quad (3)$$

where B is the blur score, $\nabla^2 I$ is the Laplacian of the grey-level image I, H is the entropy in bits, p_k is the relative frequency of histogram bin k, ρ is the edge density and C is the scene complexity. The constants 0.6, 0.4, 6.0 and 0.15 weight and normalise the two terms; they are engineering defaults and were not tuned (Section VI-E).

Mean luminance classes the scene as night, dusk, daylight or glare, and an object-density estimate is taken from the lower 70% of the frame, treated as the road region.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

D. Feasibility Pruning and Dynamic Weights

A model is viable if it fits in free memory and its estimated latency does not exceed the SLA latency by more than a tolerance, which is 20% in the service's decision engine. The tolerance keeps the candidate set from becoming empty under transient load, but it also admits models that would violate the SLA. The experiments did not use 20%: the benchmark applied the SLA bounds strictly and the ablation used 15% (Section VI-B), and none of these values is justified by experiment (Section VI-E):

$$\mathrm{RAM}(M_i) \le 0.95 \cdot \mathrm{RAM}_{\mathrm{avail}},\ L_{\mathrm{est}}(M_i) \le 1.20 \cdot L_{\mathrm{SLA}} \quad (4)$$

If no model is viable, the model with the lowest estimated latency is used as an emergency fallback. The four criteria

are accuracy (baseline mAP of the model, benefit), latency (estimated latency, cost), memory (runtime footprint, cost) and robustness (a per-model robustness score, benefit), with base weights 0.35, 0.35, 0.20 and 0.10. The model attributes are read from the registry manifest. The registry stores a baseline mAP@50 (IoU 0.50) of 0.369 for Nano, 0.448 for Balanced and 0.531 for Heavy. These are catalogue reference values that follow published benchmark reports for the YOLOv8 and RT-DETR detectors; they were not re-measured on the traffic data of this study. The robustness scores (0.65, 0.82 and 0.96) are assigned architectural scores and not measurements: they express the assumed tolerance of each model to optical degradation, lowest for the compact convolutional Nano model and highest for the transformer-based Heavy model. The base weights and the increases in Table I are engineering defaults chosen by the developer; their selection is not documented. The weights are increased by the amounts in Table I when the observed conditions cross the stated thresholds, and are then renormalised so that they sum to one.

TABLE I

Condition	Threshold	Weight increase
High blur	blur score < 40	Robustness +0.15

DYNAMIC WEIGHT ADAPTATION RULES

E. TOPSIS Ranking

The viable models are ranked by TOPSIS [25]. Each decision value x_{ij} (model i, criterion j) is vector-normalised and weighted by w_j. The ideal solution A^+ takes the best weighted value of every criterion and the anti-ideal solution A^- the worst:

$$v_{ij} = w_j \cdot x_{ij} \,/\, \sqrt{(\Sigma_k\ x^{\,2}_{kj})} \quad (5)$$

The closeness of model i to the ideal solution is then

$$C(i) = d^-(i) \,/\, (\, d^+(i) + d^-(i)\,) \quad (6)$$

where $d^+(i)$ and $d^-(i)$ are the Euclidean distances of model i to A^+ and A^-. The model with the largest $C(i)$ becomes the proposal.

F. Anti-Flapping Switch Control

A proposal that differs from the incumbent model is executed only if the incumbent has been resident for at least 3.0 s, at least 2.5 s have passed since the previous switch, and the score gain exceeds a margin that includes a switching cost that grows with the change in memory footprint:

$$\Delta S \ge 0.08 + C_{\mathrm{switch}} \quad (7)$$

$$C_{\mathrm{switch}} = 0.02 + \min(0.05, |\Delta Mem|/500 \cdot 0.05) \quad (8)$$

Every external dependency has a degraded mode (SQLite, in-memory broker, polling, offline cache). Experiments are reproducible from versioned configuration files: each run writes JSON summaries and CSV time series from which the tables in this paper were taken. The backend test suite of 249 items (decision engine, characterisation operators, fallbacks and fault injection) passed in the final run with no failures, skips or errors, in 1334.59 s under Pytest 9.1 with pytest-asyncio on Python 3.13; the front-end suite of 10 Vitest tests also passed. Passing tests show that the implementation behaves as specified; they are not a measure of code coverage or of the quality of the decisions. *Overhead.* Timing was measured on the benchmark host over repeated runs. Over 1,000 decisions the TOPSIS solver took a mean of 0.035 ms (standard deviation 0.007 ms, range 0.033–0.105 ms), and parsing the decision context, shifting the weights and writing the audit record took a further 0.261 ms (standard deviation 0.045 ms). Over 500 frames of 640×360 pixels the full 14-metric characterisation took a mean of 52.82 ms (standard deviation 2.00 ms, range 48.83–56.96 ms), whereas the fast subset (Laplacian variance, mean luminance and Immerkær noise) took 4.12 ms (standard deviation 0.65 ms). The latencies in Section VI are inference and switching latency only and exclude characterisation. The decision cost is negligible, but the full characterisation is as slow as the Balanced model's inference (46.8 ms) and would use up most of the 55 ms bound if run on every frame, so a deployment would run the fast subset on each frame, or the full pipeline at a lower rate; this was not evaluated.

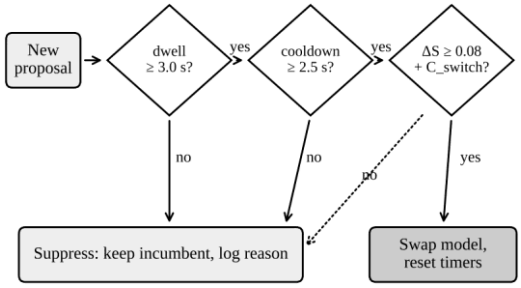


Fig. 2. Decision flow of the anti-flapping switch controller.

Each execution or suppression is logged with its reason. A closed-loop controller also re-evaluates the decision when one of nine trigger rules fires, for example a latency SLA breach, CPU overload or input-quality degradation.

IV. IMPLEMENTATION AND DATA PIPELINE

The framework is implemented in Python as an asynchronous FastAPI service with SQLAlchemy persistence (PostgreSQL 16, with an automatic SQLite fallback) and Redis publish/subscribe (with an in-memory fallback). An edge agent samples telemetry with psutil, runs inference through ONNX Runtime, keeps an offline cache and can

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

inject failures such as camera, backend or network loss. A React dashboard receives live data over WebSocket, with HTTP polling as a fallback. The services are packaged with Docker Compose.

Seen as a data pipeline, the system ingests two streams, video frames and device telemetry. It assesses frame quality, transforms the raw values into normalised feature vectors, stores decisions, switch logs and metrics in a relational schema, and delivers results to the dashboard.

V. EXPERIMENTAL SETUP

All experiments ran on one CPU-only host (four cores, eight threads, 15.6 GB RAM, ONNX Runtime CPU provider). The data are a 32-frame scenario identified as ua-detrac-v1 (after UA-DETRAC [26]) with four phases of eight frames. Each phase is defined by a scripted profile of image characteristics and of device conditions (Table III), and the device conditions of a phase are injected and not measured on the host. Phase 3 degrades the images while the hardware is not stressed, which a resource-only policy cannot detect, and Phase 4 stresses the hardware. In the benchmark the SLA was latency ≤ 55 ms, mAP ≥ 0.40 and power ≤ 15 W; the ablation used a looser SLA (Section VI-B). Six strategies were compared: A, fixed Nano; B, fixed Balanced; C, fixed Heavy; D, resource-aware selection; E, data and resource selection; F, the full framework. A separate ablation enabled the information sources in five steps: device only (Exp 1), data only (Exp 2), device and data (Exp 3), plus SLAs without the multi-objective decision (Exp 4), and the full multi-objective configuration with guardrails (Exp 5). Tables II and III summarise the setup.

Metrics. The study reports mean and P95 latency, RAM footprint, the number of model switches, a modelled mAP, selection accuracy (ablation) and, for the benchmark, SLA compliance and a composite efficiency index. A frame satisfies the SLA if its cycle latency (inference plus switching penalty) is at most 55 ms and its modelled mAP is at least 0.40; SLA compliance is the share of such frames. The efficiency index is $0.1 \cdot (F1 \cdot FPS)/((RAM/1024) \cdot P \cdot (L/1000))$, with RAM in MB, power P in W and latency L in ms. Its F1 is not measured: it is derived from the modelled mAP as precision $\min(0.99, 1.25 \cdot mAP)$ and recall $\min(0.99, 1.05 \cdot mAP)$, and power is the simulated model below. The index therefore carries no information beyond modelled mAP, simulated power, latency and memory, and it rewards cheap models (Section VI-A). The ablation applies its own rule: a frame is compliant if latency ≤ 105 ms, mAP ≥ 0.35, RAM ≤ 500 MB and simulated power ≤ 20 W, so its compliance values also depend on the simulated power model and are not comparable with the benchmark values.

Modelled mAP. The mAP values are not measured against ground-truth annotations on the frames. The baseline mAP@50 of each model is read from the registry (Section III-D), and the realised value of a frame is the baseline multiplied by degradation factors and clamped to [0.15, 0.98]: under blur below 40 the factor is 0.62, 0.82 and 0.94 for Nano, Balanced and Heavy; under complexity above 0.70 it is 0.75, 0.88 and 1.00; and under noise above 8.0 it is 0.92 for all models. The values therefore reflect the assumed robustness of each model and not detection quality on real frames. The factors are developer-chosen assumptions that encode qualitative expectations: a compact convolutional backbone loses most under blur and crowding, whereas a set-prediction detector without non-maximum suppression is least affected by crowding. They were not fitted to measurements.

Selection accuracy. Selection accuracy is the share of frames in which the chosen model equals a rule-based reference model m^*:

$$\mathrm{Acc} = (100 / N) \cdot \Sigma_{t=1\ldots N}\, 1[\, m_t = m^* \,] \qquad (9)$$

The reference is the first rule that applies: if SoC temperature exceeds 76 °C or CPU load exceeds 82%, Balanced (or Nano); else if blur is below 42, Heavy; else if complexity exceeds 0.65, Balanced; else if complexity is below 0.38 and blur is above 76, Nano. The reference is hand-written with thresholds close to those of the weight rules, so selection accuracy measures agreement with a plausible rule and not optimality, and it is partly circular. The case in which no rule applies is not specified.

Power model (simulated). Power is $P_t = P_{base}(m_t) \cdot [1 + \max(0, (CPU_t - 40)/150)]$, with base powers of 2.8, 7.2 and 19.5 W for Nano, Balanced and Heavy, and the switching penalty added to the cycle latency is 45, 95 and 220 ms. No power was measured.

Model precision. Nano is INT8, Balanced is FP16 and Heavy is FP32. On a CPU, FP16 is often not faster than FP32, so differences between models cannot be attributed to size or to precision alone.

TABLE II
EXPERIMENTAL SETUP

Item	Value
Host	4 cores / 8 threads, 15.6 GB RAM, no accelerator
Runtime	ONNX Runtime CPU provider
Model pool	YOLOv8n INT8, YOLOv8s FP16, RT-DETR FP32
Data	32-frame scenario ua-detrac-v1, 4 phases
SLA (benchmark)	latency ≤ 55 ms, mAP ≥ 0.40, power ≤ 15 W
SLA (ablation)	latency ≤ 105 ms, mAP ≥ 0.35, power ≤ 20 W
Devices	Simulated profiles; host is the only real device
Base weights	0.35 / 0.35 / 0.20 / 0.10
Repetitions	One run per configuration

TABLE III
SCENARIO PHASES (8 FRAMES EACH)

Phase	Image profile	Device profile
1 Clear, free flow	blur 85–95, lum. 160–190, 2–5 vehicles	CPU 20–32%, 48–52 °C
2 Rush hour	blur 70–82, complexity 0.75–0.92, 18–32 vehicles	CPU 42–58%, 58–64 °C
3 Adverse weather	blur 20–38, lum. 30–65, noise 9–16	CPU 35–48%, 54–59 °C (not stressed)
4 Thermal stress	blur 65–78, complexity 0.45–0.65	CPU 88–96%, 78–84.5 °C

VI. RESULTS AND DISCUSSION

A. Benchmark of Six Strategies

Table IV and Fig. 3 give the results. Strategy A is fastest,

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

but its modelled mAP of 0.303 is below the 0.40 requirement. Strategy C has the highest modelled mAP (0.513), but its 120.8 ms mean latency is far above the 55 ms bound. Strategy B meets the mAP bound and the mean latency bound, but its P95 latency of 75.0 ms exceeds 55 ms. Strategy D makes five switches at a cost of 675 ms and Strategy E makes two. Strategy F makes no switch and equals Strategy B in every reported value.

TABLE IV
SIX-STRATEGY BENCHMARK (32 FRAMES, CPU HOST)

Strat.	mAP (mod.)	Mean lat. (ms)	P95 (ms)	RAM (MB)	Sw.	SLA (%)
A Light	0.303	18.1	29.1	48.0	0	0.0
B Medium	0.405	46.8	75.0	135.0	0	25.0
C Heavy	0.513	120.8	193.5	320.0	0	0.0
D Resource	0.447	69.7	98.6	217.0	5	0.0
E Data+Res.	0.432	60.0	98.6	184.0	2	0.0
F Full	0.405	46.8	75.0	135.0	0	25.0

Sw. = total model switches; SLA = share of frames with latency ≤ 55 ms and modelled mAP ≥ 0.40. mAP (mod.) = modelled mAP, not measured against annotations.

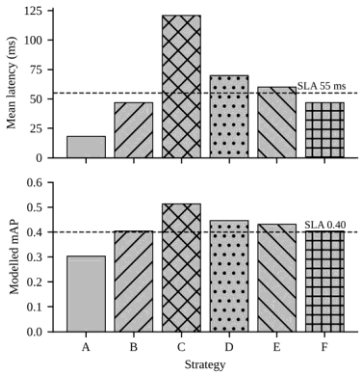


Fig. 3. Mean latency and modelled mAP of the six strategies. Dashed lines mark the SLA bounds.

Relative to Strategy D, Strategy F lowers mean latency by 32.9% (Strategy E by 13.9%) and avoids five switches; relative to Strategy C, latency falls by 61.3% at a lower modelled mAP (0.405 against 0.513). The efficiency index of Strategy F is 24.0 against 4.6 for Strategy D (+425.6% as reported by the harness, +421.7% when recomputed from the rounded values), but Strategy A has the highest index (812.3) while failing the accuracy requirement, so the index must be read together with SLA compliance. Strategies B and F satisfied the SLA on 8 of 32 frames. By the degradation factors, the modelled mAP of Balanced falls below 0.40 in Phase 2 (0.448×0.88 = 0.394) and Phase 3 (0.448×0.82×0.92 = 0.338), and its latency exceeds 55 ms in Phase 4 (72–77 ms), which leaves Phase 1 as the only compliant phase.

Strategy F is identical to Strategy B for a structural reason. Inspection of the per-frame decision log shows that Strategy F selected the Balanced model on all 32 frames, because hard-constraint pruning removed the other two models on every frame: Nano was removed because its catalogue mAP (0.369) is below the SLA floor of 0.40, and Heavy because its catalogue latency (98.6 ms) is above the 55 ms ceiling. With one viable model there was never a proposal for the switch controller to suppress, so the benchmark did not exercise TOPSIS ranking or the guardrails at all, and it cannot show a benefit of adaptive selection over the best fixed model. The harness advances a simulated clock by the cycle latency of each frame, so the whole sequence lasts 1.498 s, shorter than the 3.0 s dwell time; if the guardrail timers follow this clock, no switch could be approved in a benchmark run in any case.

B. Ablation of Information Sources

Table V and Fig. 4 give the ablation results, judged against the ablation's own SLA (latency ceiling 105 ms, accuracy floor 0.35). Mean latency falls from 73.4 ms with device information alone (Exp 1) to 70.6, 60.2 and 55.2 ms as data characteristics and SLAs are added, and to 44.7 ms for the full configuration (Exp 5), and SLA compliance is highest (100%) for Exp 5. RAM falls from 239.4 MB to 148.3 MB. The number of switches does not follow this trend: Exp 1 and Exp 4 make five switches, Exp 5 makes three and Exp 2 and Exp 3 make two.

TABLE V
ABLATION OF INFORMATION SOURCES

Exp.	Mean lat. (ms)	P95 (ms)	RAM (MB)	Sw.	Sel. acc. (%)	SLA (%)
1 Device	73.4	100.2	239.4	5	37.5	93.8
2 Data	70.6	102.7	216.3	2	75.0	87.5
3 Dev.+Data	60.2	102.7	194.5	2	50.0	87.5
4 +SLAs	55.2	101.8	140.5	5	25.0	65.6
5 Full MOO	44.7	98.6	148.3	3	75.0	100.0

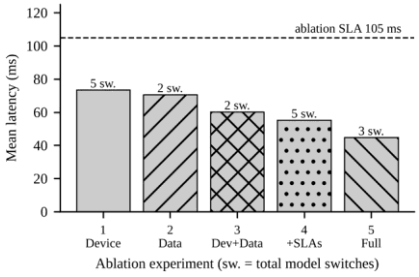


Fig. 4. Mean latency and total switches in the five ablation experiments.

The benchmark and the ablation use different configurations, which explains why Strategy F (46.8 ms, 0 switches) and Exp 5 (44.7 ms, 3 switches) differ, and the two tables must not be compared. First, the ablation SLA is looser: latency ceiling 105 ms (120.75 ms with the 15% tolerance) and accuracy floor 0.35 (0.2975), against 55 ms and 0.40 applied strictly in the benchmark. Heavy (98.6 ms) is therefore viable in the ablation. In Phase 3 blur lowers the effective accuracy of Nano to 0.369×0.62 = 0.228, below the floor, and the ranking then selected Heavy. Second, the ablation uses five criteria (accuracy, data suitability, resource headroom, latency, power) with base weights 0.25/0.20/0.15/0.20/0.20, and not the four criteria of Section III. Third, the time base differs: the benchmark clock advances by each frame's latency (1.498 s for the sequence), whereas the ablation clock advances by a fixed 1.0 s per frame, so each phase lasts 8.0 s. The ablation runner enforced a 3.0 s cooldown and a 0.08 margin and had no separate dwell check, so the switches at frames 8, 16 and 24 were approved. In Exp 1–3 the violations were frames in

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

which Heavy exceeded the 20 W power bound (or, in one frame, the 105 ms latency bound); Exp 4 deployed Nano in Phase 3, where its modelled mAP of 0.229 is below the floor for eight frames; and Exp 5 was compliant on all 32 frames. Exp 5 shows that switching occurs when the SLA admits several models and time passes, but it did not use the benchmark's dwell and cooldown values, and its latencies are not comparable with the benchmark's.

Selection accuracy is not monotonic (37.5, 75.0, 50.0, 25.0, 75.0%): data-only selection (Exp 2) equals the full configuration, and adding device information or SLAs to it lowers agreement with the reference rule while lowering latency. Because the reference rule is itself hand-written (Section V), these percentages show agreement with that rule and not better decisions. Latency decreases steadily as information sources are added, which is consistent with the premise that data characteristics carry information that resource monitoring alone lacks, as content-aware systems also assume [6]. The ablation is a single run on 32 frames, so the size of the effect and its dependence on the sequence are unknown, and no statistical significance is claimed.

C. Limitations

Scale and realism. The evaluation is one 32-frame scripted scenario and one run per configuration, with no standard deviations. The device conditions are injected, and the mAP, selection reference and power are model-based, so the study evaluates the decision logic under designed conditions and not detection performance on real traffic video. Several full UA-DETRAC sequences covering day, night, rain and dense traffic, with real annotations and repeated runs, are needed before conclusions can be drawn. Phase 3 and Phase 4 were designed to expose a resource-only policy, which favours the proposed method.

Main result. In the benchmark only one model satisfied the SLA constraints on every frame, so the full framework could not differ from the fixed Balanced model. The ablation did switch, but under a looser SLA, a five-criterion configuration, a different clock and no dwell check. A single experiment in which the same configuration, SLA and clock are used for all strategies, and the SLA admits several models in different phases, is needed.

Design constants. The base weights, the weight increases, the complexity constants, the 0.08 margin, the switching-cost formula, the dwell and cooldown times and the latency tolerance (strict, 15% or 20% depending on the runner) are defaults whose selection was not documented, and no sensitivity analysis was run. A tolerance admits models that may violate the SLA.

Hardware. The host has no accelerator, and the Jetson and Raspberry Pi profiles are simulated, so the results do not describe those boards. A measurement of latency and power on at least one real board is needed. The Heavy model fails the latency bound on CPU, as is expected for a transformer-based detector without GPU execution.

VII. CONCLUSION AND FUTURE WORK

This paper described an adaptive edge-intelligence framework that selects among three ONNX detectors from device resources, frame characteristics and SLAs, using constraint pruning, adaptive weights, TOPSIS ranking and a hysteresis controller. In a preliminary CPU-only evaluation it had lower latency than a resource-only policy and avoided its switching, and mean latency fell as information sources were added in the ablation. It did not outperform the best fixed model on the benchmark sequence, where the SLA constraints left only one viable model, and the ablation that did switch used a looser SLA, a different configuration and a different clock, so the evaluation does not yet show that adaptive selection helps. Future work should use several full UA-DETRAC sequences with real annotations, repeated runs and standard deviations, scale or justify the guardrail timers, run a sensitivity analysis on the design constants, reduce the characterisation cost, and test on a real Jetson or Raspberry Pi with measured power.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," arXiv:1905.10083, May 2019.

[3] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 2704–2713.

[4] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy, "MCDNN: An approximation-based execution framework for deep stream processing under resource constraints," in *Proc. ACM Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2016, doi: 10.1145/2906388.2906396.

[5] B. Fang, X. Zeng, and M. Zhang, "NestDNN: Resource-aware multi-tenant on-device deep learning for continuous mobile vision," in *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2018, doi: 10.1145/3241539.3241559.

[6] R. Xu *et al.*, "ApproxDet: Content and contention-aware approximate object detection for mobiles," arXiv:2010.10754, Oct. 2020.

[7] J. Jiang, G. Ananthanarayanan, P. Bodík, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proc. ACM SIGCOMM*, 2018.

[8] R. Bhardwaj *et al.*, "Ekya: Continuous learning of video analytics models on edge compute servers," in *Proc. 19th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, Renton, WA, USA, Apr. 2022, pp. 119–135.

[9] M. Khani *et al.*, "RECL: Responsive resource-efficient continuous learning for video analytics," in *Proc. 20th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, Boston, MA, USA, Apr. 2023.

[10] S. Liu, B. Guo, K. Ma, Z. Yu, and J. Du, "AdaSpring: Context-adaptive and runtime-evolutionary deep model compression for mobile applications," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 5, no. 1, Mar. 2021, doi: 10.1145/3448125.

[11] M. Zhang, F. Wang, and J. Liu, "CASVA: Configuration-adaptive streaming for live video analytics," in *Proc. IEEE INFOCOM*, 2022.

[12] A. Rahmanian, A. Ali-Eldin, S. K. Tesfatsion, B. Skubic, H. Gustafsson, P. Shenoy, and E. Elmroth, "RAVAS: Interference-aware model selection and resource allocation for live edge video analytics," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Wilmington, DE, USA, 2023, pp. 27–39, doi: 10.1145/3583740.3628443.

[13] X. Dai, Z. Zhang, P. Yang, Y. Xu, X. Liu, and J. C. S. Lui, "AxiomVision: Accuracy-guaranteed adaptive visual model selection for perspective-aware video analytics," arXiv:2407.20124, Jul. 2024.

[14] L. Wang *et al.*, "EdgeMA: Model adaptation system for real-time

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

video analytics on edge devices," arXiv:2308.08717, Aug. 2023.

[15] J. Wu *et al.*, "Efficient detection framework adaptation for edge computing: A plug-and-play neural network toolbox enabling edge deployment," arXiv:2412.18230, Dec. 2024.

[16] P. Wang, L. Huang, W. Sun, Y. Yang, D. Niyato, and D. O. Wu, "Adaptive inference acceleration with fine-grained model partitioning for mobile edge intelligence," *IEEE Trans. Mobile Comput.*, vol. 25, no. 9, pp. 13623–13637, Sep. 2026, doi: 10.1109/TMC.2026.3675369.

[17] ONNX Runtime developers, "ONNX Runtime: Cross-platform, high performance ML inferencing and training accelerator." [Online]. Available: https://onnxruntime.ai

[18] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[19] Y. Zhao *et al.*, "DETRs beat YOLOs on real-time object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 16965–16974.

[20] J. L. Pech-Pacheco, G. Cristóbal, J. Chamorro-Martínez, and J. Fernández-Valdivia, "Diatom autofocusing in brightfield microscopy: A comparative study," in *Proc. 15th Int. Conf. Pattern Recognit. (ICPR)*, vol. 3, 2000, pp. 314–317.

[21] J. Immerkær, "Fast noise variance estimation," *Comput. Vis. Image Underst.*, vol. 64, no. 2, pp. 300–302, 1996.

[22] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[23] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[24] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.

[25] C. L. Hwang and K. Yoon, *Multiple Attribute Decision Making: Methods and Applications*. Berlin, Germany: Springer-Verlag, 1981.

[26] L. Wen *et al.*, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," arXiv:1511.04136, 2015 (rev. 2020).

(This work is licensed under a Creative Commons Attribution 4.0 International License.)