Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Deadline-Aware Hardening of Real-Time Object Detection Against Candidate-Inflation Latency Attacks

Salah Gontara

salah.gontara@polytecsousse.tn

Ecole Polytechnique de Sousse

Selem Trabelsi

Ecole Polytechnique de Sousse

Khaled Ben Khalifa

University of Monastir

Research Article

Keywords: real-time object detection, latency attacks, non-maximum suppression, deadline integrity, edge computing, adversarial machine learning

Posted Date: September 17th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10938635/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Deadline-Aware Hardening of Real-Time Object Detection Against Candidate-Inflation Latency Attacks

Salah Gontara[1*], Selem Trabelsi[1,2] and Khaled Ben Khalifa[3,4]

[1*]Ecole Polytechnique de Sousse, Sousse, 4021, Tunisia.
[2]Systems Integration & Emerging Energies Laboratory, LR21ES14, National Engineering School of Sfax (ENIS), University of Sfax, Sfax, 3038, Tunisia.
[3]Technology and Medical Imaging Laboratory, Faculty of Medicine Monastir, University of Monastir, Monastir, 5019, Tunisia.
[4]Higher Institute of Applied Science and Technology of Sousse, Sousse, 4003, Tunisia.

*Corresponding author(s). E-mail(s): salah.gontara@polytecsousse.tn;

Abstract

Real-time object detectors are usually reported by mean latency or frame rate. An availability failure, however, occurs when a single response misses the application's deadline. Candidate-inflation latency attacks drive the number of boxes entering non-maximum suppression (NMS) far beyond its clean range. Recent evaluations report that realistic hardware, representations and optimized suppression reduce the severity of such attacks. Reduced severity is not a bound: work remains attacker-controlled, and prior studies do not jointly report deadline-relative tail latency and deadline-miss ratio for a defended serving path. We therefore admit at most $\boldsymbol{K}$ candidates to NMS. The selection operation is standard; new is its use as a deployment-selected bound on attacker-controlled work, calibrated against a declared deadline. It needs no retraining and no weight change, so it applies to an already promoted artifact. Replaying identical attacked candidate arrays through three suppression backends shows the attack to be strongly implementation dependent, yet under our principal attack budget no uncapped backend meets a 30-Hz frame deadline. The bound holds across two detector geometries of differing anchor count, three backends, a second evaluation corpus and a thermally throttled edge board, while the runtime's shipped candidate ceiling meets the suppression screen on the GPU backend only. The accuracy the bound costs is measured rather than assumed. Per-stage instrumentation then shows that, once candidate handling is bounded, the largest attack-associated increment is decode, not suppression. Bounding suppression work is necessary for deadline integrity but not sufficient.

Keywords: real-time object detection, latency attacks, non-maximum suppression, deadline integrity, edge computing, adversarial machine learning

1 Introduction

A detection is useful only if it arrives in time, so real-time vision research weighs algorithmic quality against the target platform, from edge-hardware optimization and start-up latency to power/performance trade-offs [1–3]. The verdict often turns on the deployment rather than the model: thread concurrency alone moves inference throughput to a board-specific saturation point

[4]. Streaming perception folds latency into accuracy [5], and latency-SLO serving systems schedule against a declared deadline [6]. Our question is not what delay costs in accuracy, but how often a deadline is missed.

The deadline every measurement here is scored against is the frame period of the camera the detector consumes, and the reasoning is worth making explicit. A detector in a video pipeline is a periodic task: frames arrive at a fixed rate, and each must finish before its successor arrives or the pipeline falls progressively further behind. Real-time practice specifies such a task by a period and a relative deadline, and industry surveys report the deadline commonly set equal to the period [7]. Under that implicit-deadline model the detector must finish one frame within one frame interval. Thirty hertz is a standard capture rate for the surveillance and driving video this class of detector is deployed on, which fixes the deadline at $D = 1/30$ s $= 33.3$ ms. Exceeding it does not merely return a slower answer: streaming perception shows that a result delivered after the frame period describes a scene the next frame has already replaced, so the output is stale [5]. Thirty frames per second is a common real-time target for detection [8] and for the edge deployments studied at this venue [1–4]. Looser budgets exist; a full driving pipeline is commonly specified at 100 ms end to end [9]. The value is therefore a declared parameter of this deployment, not a definition of real time, and Eq. (6) lets a deployment with a different period select its own. How it must be reported does not vary with the number. A mean is insufficient, since a small fraction of requests may exceed the deadline while mean latency looks healthy. Tails have long been decisive in deployed systems [10] and inference benchmarks bound latency at a percentile rather than on average [11]. We therefore define availability against the deadline and report the tail with the deadline-miss ratio.

Availability is also an adversarial concern: since sponge examples [12], a line of work has optimized inputs for computational cost rather than for a wrong class, and against detectors it targets the work NMS performs [13, 14]. Across that family the amount of work is input-controlled, which is what a deadline cannot tolerate. The attacks are also strongly implementation dependent: EVADE reports that published NMS attacks weaken under realistic hardware, formats, quantization and optimized suppression [15], and architecture-level analysis agrees [16]. An attack measured only with a slow reference implementation is not evidence of a real-time vulnerability; as we show, a fast one is not a work bound either.

This paper proposes no new attack. It asks how an established attack class should be evaluated and bounded in a deployed real-time detector. Its contributions are:

1. *Deadline-oriented evaluation.* We score the attack by client-scope p95 latency and the deadline-miss ratio, keeping the client, worker and suppression scopes separate rather than pooled.
2. *Implementation-aware characterization.* We replay one identical set of attacked candidate arrays through three suppression backends, which separates what the implementation contributes from what the attack does.
3. *Bounded candidate admission.* We admit at most K candidates to NMS. The selection operation is standard; its use as a deployment-selected bound on attacker-controlled work is not, and we measure the accuracy it costs.
4. *Deadline-calibrated selection of K.* We give a rule that picks K from measured tail latency and a stated accuracy tolerance, so a deployment with a different frame period can select its own.
5. *End-to-end validation and an acceptance criterion.* We serve the defended artifact against a matched same-format control, and express the resulting requirements as a deadline-aware condition on promotion.

2 Related work

The attacks this paper reproduces are established: sponge examples introduced inputs optimized for computational cost [12], Daedalus disrupted NMS [17], and Phantom Sponges and Overload made the candidate count entering suppression the objective [13, 14]. The idea carries to other perception pipelines [18, 19] and beyond detection entirely. In all of them the amount of work is input-controlled, which is the property this paper bounds.

The responses to it fall into five roles, and the division by role matters more than the chronology.

Attack realism asks whether the threat survives deployment: EVADE evaluates published NMS attacks across hardware, representation, quantization and optimized suppression, and reports their impact sharply reduced [15]. *Architectural mitigation* moves the work instead of removing it, Chen et al. selecting an NMS implementation by platform behavior [16]. *Retraining-based defense* changes the detector itself [20]. *NMS-free redesign* deletes the stage [21, 22], which does not help an already promoted artifact. The fifth role is this paper's: a *bounded-work defense*, fixing an upper limit on the quantity the attacker controls.

Our position with respect to EVADE is not a disagreement, and it is the paper's central positioning claim. Realistic deployment can reduce observed attack severity. Deployment-dependent mitigation is not, however, equivalent to a declared upper bound on attacker-controlled NMS work. A faster backend lowers the constant; the count stays the attacker's.

Bounding candidate admission is not an unconsidered idea; it is a dismissed one. Phantom Sponges raises it and rejects it because limiting the candidates passed to NMS "may also serve the purposes of the attacker by compromising the model's integrity" [13], alongside a timeout, which plainly does serve the attacker by dropping the frame. That objection is an empirical claim about accuracy, never measured at an operating point chosen against a deadline; measuring it is one of this paper's contributions, and the cost is 0.00064 AP@[.50 : .95].

3 System model and experimental method

3.1 Measured system

Figure 1 shows the testbed and carries its configuration. The upper row is the online path: the client's timer defines client scope, and nginx spreads requests over two workers, each holding one RTX 4090 and running YOLOv8x-P2 at 1280-pixel input on a TensorRT fp16 serving path. The lower row is offline replay, where one worker's captured pre-NMS candidate arrays feed three suppression backends and the edge board, so those arms differ only in implementation. The served operating point is $\tau = 0.30$, NMS IoU 0.65, at most 300 detections and the deployed cap $K = 1024$; the graph emits 136,000 anchors. Attacks use 48 source images at seed 1337, and each served arm is 144 requests. VisDrone2019-DET is the trained and evaluated aerial corpus [23]; BDD100K is an evaluation distribution only, the model being neither trained nor adapted on it, though attacks are crafted from it for §6.2 [24]. Serving is ONNX Runtime 1.19.2 on TensorRT 10.2.0; the backend comparison uses torchvision 0.20.1 with CUDA 12.4 on driver 595.84 over a 12-core Ryzen 9 9900X at the powersave governor, and the edge board runs ONNX Runtime 1.27.0.

Table 1 Accuracy provenance for the two detection sets.

Set	Scoring point	AP@[.50 : .95]
S	wide open	0.3186
S	threshold	0.2793
S	budget	0.3171
S	threshold + budget	0.2793
R	wide open	0.3529
R	threshold	0.313556
R	threshold + capped	0.312913

The held-out VisDrone test-dev AP@[.50 : .95] of the deployed detector is 0.2729, against a validation figure of 0.3540 from the package's own metrics, both scored COCO-style at confidence 0.001 with up to 1000 detections per image rather than the reference default of 100, since VisDrone images routinely carry more than 100 objects.

Scoring convention and operating point are distinct, and the paper needs the second. Table 1 holds the two detection sets involved. S is the served path's detection dump re-scored offline, giving deployment context; R replays the captured candidate arrays through this paper's own post-processing and carries every paired cap cost. Its rows are scoring points: the served $\tau = 0.30$, the 300-detection budget and the deployed $K = 1024$. The two disagree in level, recorded in the source artifact as unreconciled, so no difference is taken across them. Within S the threshold accounts for 0.0393 of the convention-to-operating-point gap and the detection budget for only 0.0015; the two are non-additive, and since the sets do not reconcile that gap is a lower bound. Every accuracy cost below is a paired difference within R, so 0.313556 is the baseline they are subtracted from and Fig. 3 is drawn against.

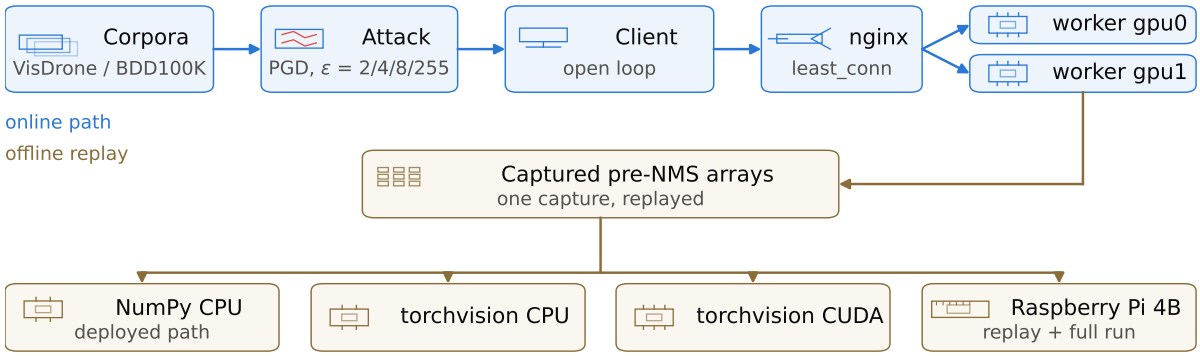


Fig. 1 The measured system and its configuration. Three paths are distinguished and only like is compared with like: the served end-to-end path (upper row), the suppression-backend replay path and the edge replay path (lower row).

3.2 Timing scopes and deadline metrics

Client-observed latency is the sum

$$L_i = T_{\text{net},i}+T_{\text{queue},i}+T_{\text{dec},i}+T_{\text{inf},i}+T_{\text{post},i}+T_{\text{ser},i}, \tag{1}$$

and the deadline-miss ratio

$$\text{DMR}(D) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[L_i > D]. \tag{2}$$

Unless stated otherwise $D = 33.3$ ms, the 30-Hz frame period of §1; we report p95 with DMR because one percentile hides the mass beyond it. The two scopes are numerically equal and logically different: D_{e2e} applies to the client-observed L_i of Eq. (1), network, queueing and serialization included, while D_{supp} screens the suppression call alone, the scope of every stage figure here. No stage allocation is claimed, $T_{\text{NMS}} \le D_{\text{supp}}$ being necessary for $L_i \le D_{\text{e2e}}$ and never sufficient, and the gap is not academic: at low load worker p95 is 28.62 ms against an open-loop client p95 of 33.48 ms, with a low-load client DMR of 0.0392 from a separate draw.

3.3 Threat model

The adversary perturbs a submitted image within an L_∞ budget and alters nothing else: not the weights, the serving binary, the cap or the server configuration. The objective is the candidate count surviving the confidence threshold before NMS:

$$\begin{aligned} \max_{\delta} \quad & N_\tau(x+\delta) \\ \text{s.t.} \quad & \|\delta\|_\infty \le \epsilon, \\ & x+\delta \in [0,1]^d. \end{aligned} \tag{3}$$

We solve Eq. (3) by projected gradient ascent on a differentiable surrogate for N_τ: with p_j the maximum class probability of anchor j, the objective is $\sum_j \sigma((p_j-\tau)/T)$, $T = 0.01$, so N_τ counts anchors, not class-expanded boxes. The update is $\delta_{t+1} = \Pi_\epsilon(\delta_t + \alpha\,\text{sign}\nabla_\delta)$, $\alpha = 0.005$, 100 steps, $\delta_0 = 0$, projected onto the ϵ-ball and $[0,1]$. Unless a budget is named, every attacked figure in this paper is the principal condition $\epsilon = 0.03$, and §4.2 varies it. It is deterministic, the seed selecting only the 48 source images, so there is no attack-seed variance. Gradients are taken white-box on the training checkpoint while the images are replayed through the exported graph, so transfer is measured rather than assumed. Sessions are the unit of uncertainty throughout: percentiles are computed within a session before being combined, and replayed requests are never independent trials. This is an input-complexity attack, not request flooding.

4 Candidate inflation and deadline failure

4.1 Candidate count controls suppression cost

Greedy NMS compares each selected candidate against the remainder, and variants that rescore



Fig. 2 A clean frame, its attacked counterpart and their amplified difference.

Table 2 Attack strength across the L_∞ budget.

ϵ (/255)	Cand. p50	Uncap. p95 (ms)	K=1024 p95 (ms)	PSNR (dB)
2	3,161	17.04	2.00	43.97
4	23,250	270.12	3.26	39.36
8	39,819	689.56	4.38	33.99

rather than discard inherit that quadratic structure: Soft-NMS is $O(N^2)$ by its authors' analysis [25]. Realized cost also depends on candidate geometry and on the implementation, so we fit no universal complexity law to it.

Figure 2 shows one such input, the frame whose attacked candidate count is closest to the arm's median, so it is typical by construction rather than chosen by eye; the panels are the letterboxed model input, padding cropped. We claim no imperceptibility, which PSNR cannot establish.

Across the corpus the p95 candidate count at the deployed threshold rises to 45,534 of the graph's M, and on an independently drawn 144-image corpus to 46,050, so the tail is not an artifact of the 48 images the headline was measured on; uncapped suppression p95 becomes 672.42 ms.

4.2 Attack strength across the perturbation budget

A single budget cannot show whether a result needs an unusually generous adversary, so we repeat the attack at $\epsilon \in \{2, 4, 8\}/255$ with source images, craft seed, graph and threshold fixed (Table 2). Latency alone would invert the reading: a smaller budget is cheaper because the attack is weaker, not because the defense is stronger.

Counts in the table are medians and latencies are 95th percentiles, against clean references of 248 candidates and 1.16 ms. The deadline behavior is not gradual: the uncapped suppression miss ratio is 0.0 at $\epsilon = 2/255$ and 1.0 at $4/255$, so the transition falls between them, at a perturbation of 39.36 dB mean PSNR and at most 4 of 255 per pixel, both measured rather than judged. The $8/255$ row anchors the sweep on the headline experiment: at $\epsilon \approx 0.031$ its uncapped and capped p95 both fall inside the principal condition's confidence intervals. The cap absorbs the entire range. Served through the deployed pool, however, these requests miss the deadline at every budget, even at $\epsilon = 2/255$ where capped and uncapped suppression both fit inside D_{supp}; that gap between stage and service is the subject of §5.3.

4.3 Implementation changes the conclusion, but does not remove the deadline violation

Backend choice substantially reduces attacked suppression latency, but no uncapped backend we tested satisfies the declared suppression screen. Following the deployment-realism concern of prior work [15, 16], we replay identical attacked candidate arrays through three suppression implementations. Over the same 48 images the suppression call's p95 is 635.35 ms on the deployed NumPy path, 177.71 ms on torchvision CPU and 49.08 ms on torchvision CUDA. CPU arms use the host clock. CUDA arms are bracketed by device synchronization after a warm-up, since an unsynchronized timing measures the kernel launch, not the work. The host-to-device copy is excluded; at 0.183 ms p95 it changes neither ordering nor verdict, so the CUDA figure understates a backend fed from the host.

The sweep separates two effects that architectural accounts conflate. NumPy against torchvision on the same CPU is implementation; CPU against CUDA is architecture. Both are large, which supports rather than dismisses the deployment dependence prior work identifies [15, 16]. Only the constant moves, however: the attacker supplies an identical count in every arm, so the quadratic term survives relocation and the CUDA call alone still exceeds D_{supp}.

Candidate inflation is not the only source of deadline misses. On one image draw the miss ratio rises from 0.0523 at 1 request/s to 0.3362 at 30, so 0.844 of it is associated with the rate rather than with input complexity: a share, not a mechanism. The budget below bounds input-dependent work and is not an overload controller.

5 Bounded candidate admission for NMS

5.1 Pre-NMS candidate budget

Top-K selection is a standard operation and we claim no novelty for it. What is proposed here is its use as a deployment-selected bound on attacker-controlled work, with the value calibrated against a declared deadline and its accuracy cost measured. Let $\mathcal{C}$ be the candidates surviving the confidence threshold and $N = |\mathcal{C}|$. Before NMS we retain only $\mathcal{C}_K = \text{TopK}(\mathcal{C}, K)$, admitting for $N > K$ the K candidates smallest under the lexicographic key $(-s_j, j)$: `argpartition` on the negated scores at index $K-1$ locates the cutoff, every candidate strictly above it is admitted, and the remaining places go to candidates equal to it in ascending index order. The tie rule is not a formality, NumPy's introselect being unstable and exact ties reachable at fp16 or quantized scores. The cut is $O(N)$ and adds no pairwise IoU comparison, the term Eq. (4) bounds.

For a serial greedy NMS implementation, no more than

$$C_{\text{pair}}(K) \le \frac{K(K-1)}{2} \tag{4}$$

pairwise IoU comparisons are required after the cap. Equation (4) is an algorithmic work bound, not a hardware worst-case execution time; establishing the latter needs architectural analysis of the specific target [26], and our backend comparison shows how far apart the two can be, so the wall-clock tail is measured on each backend.

Selection does not inherit that bound. Writing M for the graph's fixed output size (136,000 anchors here), the post-processing stages cost

$$\begin{aligned} T_{\text{post}}(M, N, K) = T_{\text{filter}}(M) + T_{\text{select}}(N, K) \\ + T_{\text{NMS}}(\min\{N, K\}), \end{aligned} \tag{5}$$

with $N \le M$. Filtering reads the whole output tensor and is a constant of the graph. Selection takes N, the attack's own objective in Eq. (3), so it remains attack dependent, but bounded by M rather than by K. The worker carries one timer across the first two terms, so §5.3 reports them jointly, rising between the matched arms from 1.76 to 2.15 ms: graph-bounded, small and observed. Every stage-scoped millisecond quoted here is T_{NMS} alone: applying the cap is itself selection work, and it is measured separately from the reported NMS timings rather than absent from them. The guarantee is narrow by construction. The count entering greedy NMS is no longer set by the attack, and the remainder is bounded by M.

5.2 Accuracy–latency trade-off

Figure 3 evaluates several values of K under three admission policies: the deployed global budget, which admits at most K candidates in total, and the per-class and per-level alternatives. Only the global policy makes the admitted total a declared quantity, so Eq. (4) applies to it alone. The other two are plotted for completeness; equal labels there are not equal admitted work, and the curves are not a ranking. The operating point $K = 1024$ reduces attacked suppression p95 to 4.28 ms. Its clean cost is the paired difference of the last two rows of Table 1, 0.00064 AP@[.50 : .95], and 0.00060 when the clean evaluation is replayed through the serving pool. Both are validation figures, while §3's headline accuracy is held out, and a cost measured on one population cannot be subtracted from an accuracy measured on another; so we repeated the sweep on test-dev with K fixed beforehand, obtaining 0.000188 over all 1610 images. The held-out cost is the smaller, so the validation figure is the conservative one to

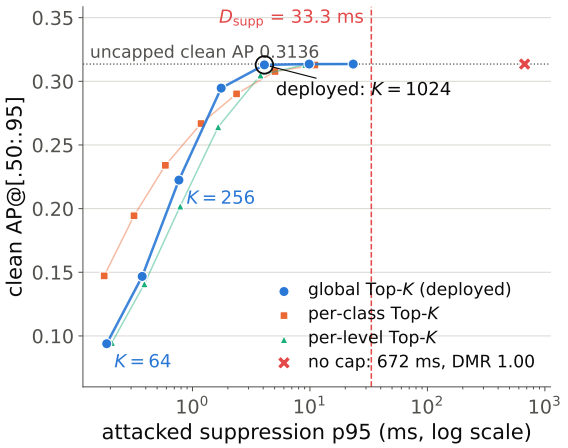


Fig. 3 Accuracy–latency frontier for the candidate budget.

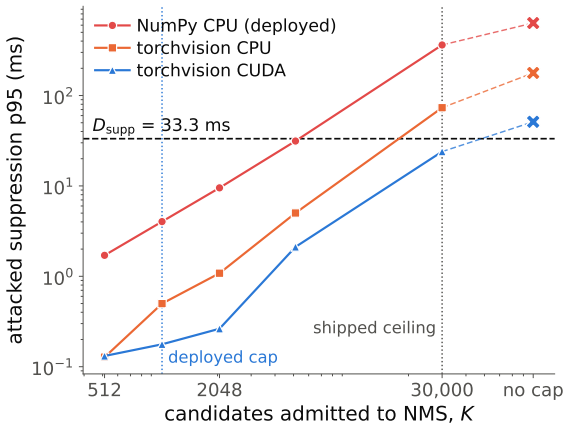


Fig. 4 Attacked suppression tail against the admitted count, on three backends.

quote, and the frontier saturates just above the deployed cap.

For the selection rule below, $U_{0.95}$ is a one-sided 95% Student-t upper confidence bound over sessions.

$$K^* = \max\{K : U_{0.95}\big(Q_{0.95}(T_{\mathrm{NMS}}(K))\big) \le D_{\mathrm{supp}} \\ \wedge\ \Delta\mathrm{AP}_K \le \delta_A\}, \tag{6}$$

evaluated over the swept grid. On the deployed NumPy path $K^* = 5,120$, the limit being a Student-t bound over three session p95 values of 48 images each; a more conservative multiplier leaves the same grid point inside the budget and the next one well outside. The deployed cap sits five times below K^*, a declared choice made for timing headroom rather than a consequence of the sweep.

Figure 4 sweeps K across all three backends on the same captured arrays. The curves stay ordered by implementation over three orders of magnitude. At the deployed cap every backend is inside D_{supp}, the worst being 4.03 ms, so the budget is not an artifact of the slow reference implementation it was chosen on. The runtime's own ceiling is where two of the three cross: at $K = 30{,}000$ attacked suppression p95 is 362.65 and 73.34 ms on the two CPU backends against 23.97 ms on CUDA. A ceiling large enough to be a memory guard is not a deadline.

5.3 Stage-local protection versus served deadline

The cap reduces client-side p95 under attack by a factor of 15.10, a median over five independent A/B sessions, yet the defended requests still miss $D_{\mathrm{e2e}} = 33.3$ ms every time. That is no contradiction: the work bound applies to suppression, and $T_{\mathrm{NMS}} \le D_{\mathrm{supp}}$ does not imply $L_{\mathrm{e2e}} \le D_{\mathrm{e2e}}$. Nor is the miss attributable to the attack alone. Attacked images must be delivered losslessly, since re-encoding at the quality this pool serves did not preserve the perturbation, so comparing them against the JPEG corpus varies two things at once. We therefore re-encoded the same 48 sources into the attacked arm's format and served all three arms through the same pool at the same cap, in ten sessions of 144 requests at $\epsilon = 2/255$.

The adversary turns out not to be necessary for the miss. Clean lossless images reach p95 66.20 ms at a deadline-miss ratio of 0.927, against 61.20 ms attacked. Those intervals overlap, so that ordering is not a ranking. The paired statistic is: taken per source image as attacked minus clean, it puts the attacked arm ahead by a median 15.02 ms. The two need not agree in sign, since one compares the arms' worst requests and the other compares each image with itself. The JPEG default reaches 31.10 ms at 0.0.

Figure 5 shows where the time goes; only worker stages are stacked, so no bar reaches a client p95. The attacked bar exceeds the clean lossless bar almost entirely in decode, which runs 1.87, 14.34 and 26.88 ms across the three arms

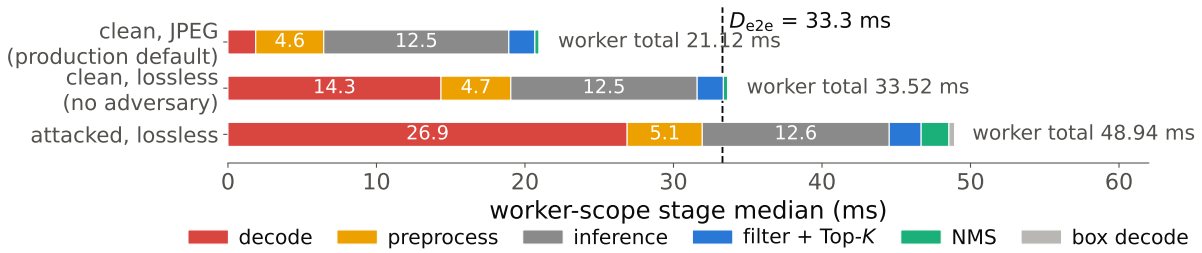


Fig. 5 Worker-scope stage medians for the three matched delivery arms. The D_{e2e} line is a reference threshold, not an end-to-end percentile: the bars stack worker stages only.

as the payload grows from 1.41 to 2.28 MB, taking worker totals from 21.12 through 33.52 to 48.94 ms. All candidate handling, the only terms the attack can inflate, costs about 2 ms more with the cap in place: suppression rises from 0.29 to 1.87 ms, and filtering plus selection, sharing one timer, from 1.76 to 2.15 ms.

The largest attack-associated increment inside the worker is therefore decode and representation, not bounded candidate handling; where image processing sits in a pipeline is itself a real-time design variable [27]. This decomposes worker medians, not client p95: the attacked arm does raise the miss ratio to 1.0, for which no binomial limit is quoted, an observed 1.0 admitting none. Without the matched control the format change would have been scored as attack effect. With it, a request no adversary has touched misses the deadline on representation alone. A bounded stage is not evidence about the service.

6 Generality and deployment behavior

6.1 A second detector geometry

The anchor grid sets the maximum candidate count, so a cap validated on one output tensor says little about another. We repeated the capture and the sweep on the same detector family without its P2 head: 33,600 anchors against 136,000, a factor of four in the quantity that sets the ceiling. $K = 1024$ holds the suppression budget on both, 3.11 ms against 4.19 ms in the same run, while the runtime's shipped ceiling of 30,000 misses on both, at 187.30 and 342.96 ms. This is suppression replay only; no accuracy claim is made for the second model, and the bound is established for the two geometries tested rather than for anchor-based detectors in general.

6.2 Second evaluation distribution

The promoted package is evaluated unchanged on BDD100K [24], a driving corpus whose clean candidate distribution is more than an order of magnitude sparser than VisDrone's, while attacked counts land at much the same level on both. Their ratio therefore rises to 1777.19, which measures how little natural load a sparse corpus carries and is not a severity ranking. The same $K = 1024$ defense nonetheless gives a client-side attacked p95 speedup of 14.04 over 144 requests per arm, and the cleanest attribution here: BDD's clean capped deadline-miss ratio is 0.0, so the attacked 1.0 on the same arm has no floor to subtract. Transfer across input distributions is what EVADE found published attacks to lack [15]; we report it for the defense.

6.3 Edge platform and thermal throttling

This experiment tests one proposition: bounded NMS work does not imply invariance of the full pipeline to platform state. The edge board evaluates the mechanism on a second architecture, cooling unmodified so that throttling is part of the measurement.

Figure 6 shows both halves. Above, the board is plainly thermally limited: temperature saturates, the first live throttle bit sets inside two minutes, and the ARM clock falls from 1.5 GHz and stays down. Below are the two inflation ratios with their session intervals against the no-change line – two workloads on one board under one thermal regime, an edge-specialist model run end to

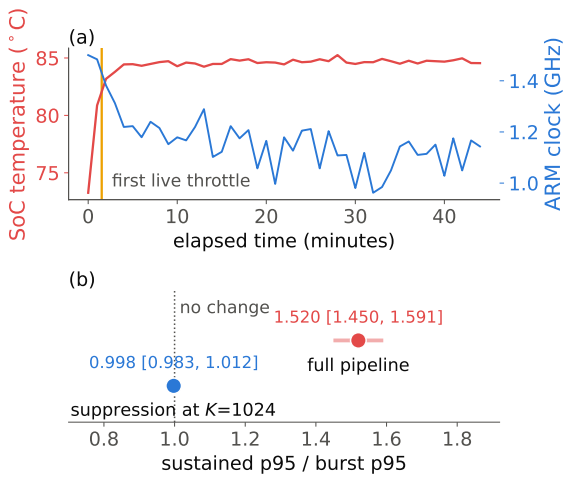


Fig. 6 The edge board under sustained load. Panel (a): platform state under sustained execution. Panel (b): full-pipeline sensitivity against bounded-NMS sensitivity to that state.

end and the deployed detector's captured candidates replayed through suppression. Full-pipeline p95 inflates by 1.52 [1.45, 1.59] relative to burst operation, while $K = 1024$ suppression p95 is unchanged at 0.998 [0.983, 1.012]: the intervals do not meet, so the bounded stage did not move while the pipeline around it did. That is empirical evidence from this platform, not a property the cap confers, since bounding candidate admission bounds pairwise comparisons and not clock, temperature or cache behavior.

7 A deadline-aware model-promotion criterion

An artifact passing accuracy and integrity checks can still violate a real-time requirement after export. We therefore propose, rather than standardize, an acceptance criterion in which deadline evidence joins promotion of package P:

$$\begin{aligned}\text{Promote}(P) = I(P) \wedge A(P) \\ \wedge R_{\text{clean}}(P) \wedge R_{\text{adv}}(P, K),\end{aligned} \tag{7}$$

with $I(P)$ provenance checks, $A(P)$ a held-out accuracy criterion, and the clean and adversarial deadline conditions

$$\begin{aligned}R_{\text{clean}} = [U_{0.95}(Q_{0.95}(L_{\text{clean}})) \le D_{\text{e2e}}] \\ \wedge [U_{0.95}(\text{DMR}_{\text{clean}}) \le \rho],\end{aligned} \tag{8}$$

$$\begin{aligned}R_{\text{adv}} = [U_{0.95}(Q_{0.95}(L_{\text{adv}})) \le D_{\text{e2e}}] \\ \wedge [U_{0.95}(\text{DMR}_{\text{adv}}) \le \rho] \\ \wedge [\Delta\text{AP}_K \le \delta_A].\end{aligned} \tag{9}$$

Both are evaluated at client scope on the exported artifact. L_{clean} covers every input representation the production API accepts, lossless included: the pool serves lossless requests (§5.3) and an adversary reaches it through the same endpoint, so a criterion evaluated only on the JPEG default would exclude the representation the attack requires. Here $U_{0.95}$ instead uses a hierarchical bootstrap over sessions and source images; no exact binomial limit is used for a miss ratio, because replayed requests over one corpus are not independent trials. We set $\rho = 0.05$ and $\delta_A = 10^{-3}$, both policy rather than results, with $\Delta\text{AP}_K = \text{AP}_{\text{uncapped}} - \text{AP}_K$ a paired difference within one detection set.

The evaluated package does not pass. The deployed $K = 1024$ package satisfies I, A and $\Delta\text{AP}_K \le \delta_A$, its held-out cap cost being 0.000188. It fails R_{adv} on the quantile conjunct: attacked served p95 is 61.20 ms against $D_{\text{e2e}} = 33.3$ ms, nearly twice the budget. The observed DMR_{adv} is 1.0 on every session, so that conjunct fails too, though we lean on the quantile, since an observation of one admits no upper bound below one. It fails R_{clean} as well, on clean lossless requests. The artifact is therefore rejected at a 30-Hz deadline with the served path, not the defense, as the binding constraint: it becomes promotable at a longer frame period, on a smaller delivery representation, or once the other terms of Eq. (1) fall.

8 Discussion

8.1 Positioning against prior work

Table 3 reports values as the cited papers give them. Heterogeneous platforms and settings limit direct comparison, so it must be read structurally: what is comparable across its rows is what each work measures. Two columns carry the gap. The first, *Timing evidence*, records whether a deadline is evaluated at all. Percentiles do appear in that literature; a declared deadline to read them against does not, and no entry reports a deadline-miss ratio. The only entry constraining a percentile as a requirement is a benchmark suite, not an attack study [11].

Table 3 Comparison with prior latency-attack and real-time deployment studies.

Work	Setting, hardware	Timing evidence	Robustness and cost	Main result and relevance
Phantom Sponges [13] (WACV 23)	Universal perturbation on YOLO; offline	Candidate counts, inference time; no tail, no deadline	Attack. Names a candidate limit, rejects it unmeasured as harming integrity	Identifies NMS load as an availability surface. We quantify the rejected limit under a deadline-calibrated budget
Overload [14] (CVPR 24)	YOLOv5, Jetson NX	Latency percentiles and ≈10× inference inflation; no declared deadline	Attack; same dismissal	Shows severe inflation on constrained hardware. We evaluate the tail against a declared deadline and report the resulting miss ratio
SlowTrack [19] (AAAI 24)	Apollo stack, LGSVL simulator	System outcome (crash rate); no latency distribution	Attack, deployment-bound	Links latency attacks to safety outcomes. We add explicit deadline and latency-distribution analysis
Chen et al. [16] (ACCV 24)	YOLOv5; V100, 1080 Ti, A10, Jetson NX	Mean per-image time; page faults 2.5M→13.3M	Defense, no retraining. Relocates NMS (async, CPU-free)	Mitigates architectural overhead but does not bound the candidate count. We quantify that count against a deadline
Underload [20] (CVPR 25)	Jetson Orin NX	Throughput under attack (13→43 FPS)	Defense; adversarial retraining	Uses retraining to suppress inflation. Our defense is weight-independent and compatible with a fixed deployed artifact
EVADE [15] (NeurIPS 25)	Hardware, format, quantization, optimized NMS	Impact after realistic deployment; limited transfer	Evaluation; deployment blunts the attack	Shows strong implementation dependence. We distinguish reduced severity from an explicit bound on attacker-controlled work
Real-time deployment work at this venue [1–4]	Edge boards, embedded accelerators	Start-up latency, throughput, power; benign load	No adversarial component	Establishes deployment-level timing constraints. We extend the framing to adversarial load
This work	YOLOv8x-P2 @1280, TensorRT fp16, 2-GPU pool; VisDrone, BDD100K (eval only), Pi 4	p95 *and* deadline-miss ratio at $D_{\mathrm{e2e}} = 33.3$ ms; client, worker, suppression scopes separated	Defense: bounded admission. No retraining; accuracy cost measured held out	Suppression severity is implementation dependent and stays attack-controlled without a deployment-selected cap. Bounded admission supplies an explicit work bound, though not end-to-end sufficiency

The second, *Robustness and cost*, records whether the attacker-controlled quantity is bounded or only made cheaper; our backend sweep prices that difference. Identical attacked arrays move from 635.35 to 49.08 ms p95 between the deployed NumPy path and torchvision CUDA. That is agreement with the deployment-realism literature, not a rebuttal. It is also where the argument turns: the faster call remains above D_{supp} with the attacker's count untouched, while the same arrays under the cap finish in 4.28 ms. The distinction is deployment-bounded versus attack-controlled work, not model versus runtime.

Two contrasts follow. The held-out integrity cost of the limit is 0.000188 AP@[.50 : .95], so Phantom Sponges' reason for setting it aside [13] does not survive measurement at a deadline-calibrated point. And the inflation ratios this family is reported in are a misleading currency: ours reaches 1777.19 on BDD100K purely because a sparse corpus carries little clean load, while the defended p95 and the deadline verdict are much the same on both. Ratios rank corpora; deadlines rank deployments.

8.2 Threats to validity

Two choices are policy, not findings. p95 is the policy percentile because it is the quantile our session counts support with a useful confidence limit; p99 would be stricter and less precisely estimated here. And K^* rests on three sessions, a weak basis for a tail statement; the deployed cap sits five times below it, so the margin absorbs that, but a deployment adopting K^* should re-estimate over more.

Re-thresholding one nine-session clean low-load draw across deadlines, the miss ratio is 0.0093 at 40 ms (25 Hz) and zero over all 2592 requests by 66.7 ms. The mass sits in a narrow band immediately above the budget: a service at its boundary, not one unable to meet a deadline, and the looser pipeline budget of §1 would contain it.

Suppression cost depends substantially on the implementation [15, 16], so the algorithmic candidate bound is the portable result; the millisecond envelope is specific to the platforms measured here. Our claim is bounded candidate admission plus measured end-to-end validation, not formal WCET [26]: candidate inflation is one attack-controlled term among several, and the others, while not adversarial here, are not small.

9 Conclusion

Candidate inflation turns the post-processing stage of NMS-based detectors into an availability surface whose severity depends strongly on the deployed suppression backend. Bounding candidate admission limits this attack-controlled work rather than merely reducing it. A cap of $K = 1024$ provides that protection at an accuracy cost of 0.00060 AP@[.50 : .95] for the evaluated detector and operating point, and it holds across two detector geometries, three suppression backends, two evaluation corpora and thermally constrained edge execution. The selection rule we give lets a deployment choose its own bound, and our criterion makes that evidence a release precondition.

The experiments also establish the boundary of the defense. Once candidate inflation is controlled, deadline violations can persist through costs outside suppression, and matched-format controls with per-stage timing isolate delivery and decoding as the next bottleneck. Bounding attack-controlled computation is therefore necessary for deadline integrity but not sufficient. Robust real-time vision needs explicit control of adversarially amplifiable work together with measurement of the remaining latency path.

Statements and declarations

The authors received no funding for this work and declare no competing interests.

Data availability

Both corpora are public and cited [23, 24]. Every headline numeric value here, including those inside figures, carries an identifier resolving against an evidence ledger that records its source artifact, the pointer into it, the sample count and a content hash. That ledger, the artifacts it cites, the figure provenance records and a script that re-resolves every claim against them are archived at doi:10.5281/zenodo.22149296; the defense, the attack objective and the deadline metrics are implemented at doi:10.5281/zenodo.22117976. The crafted image sets and captured candidate arrays, derivatives of the two corpora, are available from the corresponding author on request.

References

[1] Li, Q., Li, H., Meng, L.: A generic deep learning architecture optimization method for edge device based on start-up latency reduction. Journal of Real-Time Image Processing **21**(4), 116 (2024)

[2] Mela, J.L., Garc'ia S'anchez, C.: YOLO-based power-efficient object detection on edge devices for USVs. Journal of Real-Time Image Processing **22**(3), 108 (2025)

[3] Contoli, C., Calisti, L., Di Fabrizio, G., Kania, N., Bogliolo, A., Lattanzi, E.: A power-aware vision-based virtual sensor for real-time edge computing. Journal of Real-Time Image Processing **21**(4), 103 (2024)

[4] Costa, C., Brito, J.H.: Throughput impact of software multithreading for deep-learning inference on the AMD Kria KV260. Journal of Real-Time Image Processing **23**(2), 89 (2026)

[5] Li, M., Wang, Y.-X., Ramanan, D.: Towards streaming perception. In: Computer Vision – ECCV 2020, pp. 473–488 (2020)

[6] Shen, H., Chen, L., Jin, Y., Zhao, L., Kong, B., Philipose, M., Krishnamurthy, A., Sundaram, R.: Nexus: A GPU cluster engine for accelerating DNN-based video analysis. In: Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP), pp. 322–337 (2019)

[7] Åkesson, B., Nasri, M., Nelissen, G., Altmeyer, S., Davis, R.I.: A comprehensive survey of industry practice in real-time systems. Real-Time Systems **58**(3), 358–398 (2022)

[8] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788 (2016)

[9] Lin, S.-C., Zhang, Y., Hsu, C.-H., Skach, M., Haque, M.E., Tang, L., Mars, J.: The architectural implications of autonomous driving: Constraints and acceleration. In: Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 751–766 (2018)

[10] Dean, J., Barroso, L.A.: The tail at scale. Communications of the ACM **56**(2), 74–80 (2013)

[11] Reddi, V.J., Cheng, C., Kanter, D., *et al.*: MLPerf inference benchmark. In: ACM/IEEE International Symposium on Computer Architecture (ISCA) (2020)

[12] Shumailov, I., Zhao, Y., Bates, D., Papernot, N., Mullins, R., Anderson, R.: Sponge examples: Energy-latency attacks on neural networks. In: IEEE European Symposium on Security and Privacy (EuroS&P) (2021)

[13] Shapira, A., Zolfi, A., Demetrio, L., Biggio, B., Shabtai, A.: Phantom Sponges: Exploiting non-maximum suppression to attack deep object detectors. In: IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) (2023)

[14] Chen, E.-C., Chen, P.-Y., Chung, I.-H., Lee, C.-R.: Overload: Latency attacks on object detection for edge devices. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2024)

[15] Monteuuis, J.-P., Chen, C., Petit, J.: Latency NMS attacks: Is it real life or is it just fantasy? In: Advances in Neural Information Processing Systems (NeurIPS) (2025)

[16] Chen, E.-C., Chen, P.-Y., Chung, I.-H., Lee, C.-R.: Latency attack resilience in object detectors: Insights from computing architecture. In: Proceedings of the Asian Conference on Computer Vision (ACCV), pp. 3206–3222 (2024)

[17] Wang, D., Li, C., Wen, S., Han, Q.-L., Nepal, S., Zhang, X., Xiang, Y.: Daedalus: Breaking nonmaximum suppression in object detection via adversarial examples. IEEE Transactions on Cybernetics **52**(8), 7427–7440 (2022)

[18] Liu, H., Wu, Y., Yu, Z., Vorobeychik, Y., Zhang, N.: SlowLiDAR: Increasing the latency of lidar-based detection using adversarial examples. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023)

[19] Ma, C., Wang, N., Chen, Q.A., Shen, C.: SlowTrack: Increasing the latency of camera-based perception in autonomous driving using adversarial examples. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38 (2024)

[20] Wang, T., Wang, Z., Wang, C., Shu, Y., Deng, R., Cheng, P., Chen, J.: Can't slow me down: Learning robust and hardware-adaptive object detectors against latency attacks for edge devices. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 19230–19240 (2025)

[21] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 213–229 (2020)

[22] Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., Ding, G.: YOLOv10: Real-time end-to-end object detection. In: Advances in Neural Information Processing Systems (NeurIPS) (2024)

[23] Zhu, P., Wen, L., Du, D., Bian, X., Fan, H., Hu, Q., Ling, H.: Detection and tracking meet drones challenge. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(11), 7380–7399 (2022)

[24] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., Darrell, T.: BDD100K: A diverse driving dataset for heterogeneous multitask learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2636–2645 (2020)

[25] Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-NMS — improving object detection with one line of code. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 5562–5570 (2017)

[26] Wilhelm, R., Engblom, J., Ermedahl, A., Holsti, N., Thesing, S., Whalley, D., Bernat, G., Ferdinand, C., Heckmann, R., Mitra, T., Mueller, F., Puaut, I., Puschner, P., Staschulat, J., Stenstr"om, P.: The worst-case execution-time problem—overview of methods and survey of tools. ACM Transactions on Embedded Computing Systems **7**(3), 1–53 (2008)

[27] Benjak, J., Hofman, D., Mlinari'c, H.: Deferred demosaicking: Efficient first-person view drone video encoding. Journal of Real-Time Image Processing **22**(2), 101 (2025)