RoadSense AI: Zero-Shot versus Fine-Tuned YOLO11 for Traffic-Camera Detection, Tracking and Counting – A Reproducible Evaluation

Akter Mst Lovely
Nanjing University of Information Science and Technology (NUIST)
Nanjing, China

Mostafa Farzana
Nanjing University of Information Science and Technology (NUIST)
Nanjing, China

Iftee Shekh Iftesham
Nanjing University of Information Science and Technology (NUIST)
Nanjing, China

Tanha Asfika Jaman
Nanjing University of Information Science and Technology (NUIST)
Nanjing, China

ABSTRACT
Traffic-monitoring pipelines built on pretrained detectors are often described as accurate, robust and real-time without measurements on traffic-camera data. This paper re-evaluates RoadSense AI, a pipeline that combines YOLO11 detection or instance segmentation, multi-object tracking and virtual-line counting, after an audit that withdrew its original evaluation artefacts and its test videos of unknown provenance. On the COCO val2017 traffic classes, box AP is 40.3 for YOLO11n-seg and 54.2 for YOLO11m-seg. On the 40 UA-DETRAC test sequences, zero-shot vehicle box AP (IoU 0.50:0.95) is 56.5 and 61.9. Fine-tuning a COCO-pretrained YOLO11n detector for 12 epochs on sequence-disjoint UA-DETRAC training data raises AP on held-out training-partition sequences from 61.0 to 67.3, but not on the test set (56.8 ± 0.3 over three seeds versus 56.6 for the same detector zero-shot; every paired 95% CI includes zero). The validation gain is concentrated at camera sites that also occur in training (+9.0 points, against −2.0 elsewhere). At the unseen test sites the typical change is small (median +1.9 points over 20 sites) and is cancelled in pooled AP by large losses at four night-time or wet-road sites; fine-tuning also recalibrates confidence scores. The best tracking configuration, zero-shot YOLO11m-seg with BoT-SORT, reaches HOTA 68.6 and a line-count error of 4.0 vehicles per sequence; the original centroid tracker reaches HOTA 48.1 and under-counts by 8.2 vehicles per sequence. The full loop runs at 50 frames/s on an RTX 4060 Ti and 21 frames/s on a Ryzen 5 7600X with YOLO11n-seg. The original wall-clock speed read-out ranged from 6 to 127 'km/h' on the same frames depending on hardware. Validation data for traffic cameras should be split by camera site.

General Terms
Computer Vision, Pattern Recognition, Algorithms, Measurement, Experimentation.

Keywords
Traffic monitoring, YOLO11, UA-DETRAC, domain shift, site-disjoint validation, fine-tuning, multi-object tracking, vehicle counting, robustness, reproducibility.

1. INTRODUCTION
Camera-based traffic monitoring turns existing roadside video into vehicle counts and flow indicators without new in-road sensors [1–3]. Single-stage detectors of the YOLO family are a common building block of low-cost systems [4, 5], and they are often deployed exactly as downloaded: pretrained on COCO [6] and applied to traffic cameras without adaptation. The earlier, course-report version of the system studied here was described in exactly these terms, as accurate, robust and real-time, without measurements that could support the description.

Such properties are easy to assert and easy to measure on the wrong data. COCO accuracy says little about elevated traffic cameras, where vehicles are small, densely packed and seen from above [7]. Throughput of the network alone says little about a full loop that also decodes, tracks, renders and plots. Derived quantities have their own failure modes: an identity tracker can inflate counts, and a speed read-out in km/h has no physical meaning unless it is tied to calibrated geometry and to the timeline of the video [8, 9]. Finally, even a carefully held-out validation set can mislead when it shares scenes with the training data [10–12].

This paper examines RoadSense AI, a traffic-monitoring pipeline built on pretrained YOLO11 instance-segmentation models [5, 13], and asks three questions. (Q1) How accurate is the training-free pipeline on traffic-camera ground truth, compared with its accuracy on COCO? (Q2) How much does inexpensive fine-tuning on sequence-disjoint traffic-camera data change detection, tracking and counting, and does the gain seen during model selection carry over to new cameras? (Q3) Which parts of the pipeline, rather than the detector, limit validity and speed?

The earlier version was audited before these questions were addressed. Several of its evaluation artefacts were invalid: the confusion matrix compared predictions with themselves, the precision–recall and F1 curves were analytic functions, the speed read-out used wall-clock processing time, a colour map labelled Grad-CAM involved no gradients, and segmentation masks were rendered with incorrect geometry. Its evaluation videos also had no recorded source or licence. All of these artefacts and videos are excluded here. Every number in this paper comes from a logged experiment on COCO val2017 or on the UA-DETRAC benchmark [7], with recorded inputs, software versions and hardware.

Contributions. This is an engineering and evaluation study. It proposes no new network architecture and claims no state-of-the-art accuracy. Its contributions are:

- **Ground-truth accuracy in two domains.** Box and mask AP on the COCO traffic classes and vehicle box AP on all 40 UA-DETRAC test sequences for three YOLO11 sizes, with bootstrap confidence intervals resampled at the image level (COCO) or at the sequence and site levels (UA-DETRAC) [14].
- **A controlled fine-tuning study with a site audit.** A COCO-pretrained YOLO11n detector is fine-tuned on sequence-disjoint UA-DETRAC data with three seeds and compared with its own zero-shot baseline. The pooled validation gain does not carry over to pooled test AP; per-sequence and per-site analyses associate most of the discrepancy with camera sites shared between training and validation sequences.
- **Tracking and counting against identity ground truth.** HOTA, MOTA and IDF1 [15, 16] and virtual-line count errors for the original centroid tracker, ByteTrack [17] and BoT-SORT [18].
- **Robustness, latency and pipeline pitfalls.** Twelve deterministic degradations scored with one evaluator on one fixed COCO subset [19, 20]; a per-stage latency budget on a consumer GPU and a CPU; and measured effects of the wall-clock speed read-out, the mask-rendering error and lane drawing before detection.
- **A verified evaluation record.** Every reported metric was recomputed by an independent implementation or re-executed from stored artefacts, and inference was re-run to confirm bit-identical outputs.

Section 2 reviews related work, Section 3 defines the pipeline, Section 4 the data and protocol, Section 5 the results; Section 6 discusses implications and validity and Section 7 concludes.

2. RELATED WORK

2.1 Detection and instance segmentation

Two-stage detectors [21, 22] and Mask R-CNN [23] set the accuracy reference for boxes and masks, while SSD and the YOLO family [4, 24–27] trade accuracy for throughput. Prototype-based mask heads [28] made real-time instance segmentation practical; the same idea is used by the segmentation variants of Ultralytics YOLOv8 and YOLO11 [5, 29], whose YOLO11 backbone and neck revisions are summarised in [13]. Public weights are trained on COCO [6], whose ground-level photographs differ from elevated traffic cameras.

2.2 Traffic-camera benchmarks and tracking

UA-DETRAC provides 100 sequences (60 training, 40 test) of 960×540 traffic video at 25 frames/s with per-frame vehicle boxes, identities and ignore regions [7]; the AI City Challenge extends this setting to city-scale problems [30]. SORT [31] links detections with a Kalman filter [32] and Hungarian assignment; DeepSORT [33] adds appearance features; ByteTrack [17] also associates low-score detections; BoT-SORT [18] adds camera-motion compensation. CLEAR-MOT, IDF1 and HOTA [15, 16] require identity ground truth such as UA-DETRAC's.

2.3 Counting, speed and robustness

Counting systems usually follow a detect–track–count design: Majin et al. compared single-stage detectors before counting [34], and Ugalino et al. combined YOLOv8 with ByteTrack, rectified the view with a homography and checked counts against manual tallies [35]. Fernández Llorca et al. stress that metric speed requires calibration or known scene geometry [8], which BrnoCompSpeed provides as ground truth [9]. Robustness to common corruptions has been benchmarked for classifiers and detectors [19, 20].

2.4 Domain shift, leakage and calibration

Benchmarks carry their own visual biases, so accuracy measured on one dataset or one set of scenes overstates accuracy elsewhere [10, 36]; domain-adaptive detectors address the shift explicitly [37]. When observations are grouped, by site, subject or time, random or record-level splits leak information between training and evaluation; grouped or blocked validation is the recommended remedy [11], and leakage of this kind is a documented source of irreproducible machine-learning results [12]. Modern networks are often miscalibrated [38], which matters for any fixed threshold. Table 1 positions this study.

Table 1. Positioning of this study against closely related work

Work	Focus	Ground truth	Speed
Majin 2021 [34]	Detector choice, counting	Own counts	–
Ugalino 2026 [35]	YOLOv8 + ByteTrack flow	mAP, manual counts	Homography
Sochor 2019 [9]	Speed benchmark	Metric speed	Calibrated
Michaelis 2019 [20]	Detector robustness	COCO, Cityscapes	–
This study	**Audit; COCO→DETRAC accuracy, fine-tuning with site audit, tracking, latency**	**COCO; UA-DETRAC boxes + IDs**	**px/s proxy only**

3. PIPELINE AND DEFINITIONS

The pipeline (Figure 1) processes one frame at a time. A YOLO11 model predicts boxes, class scores and, for the segmentation variants, prototype-based instance masks at an input size of 640 pixels [5]. For COCO-pretrained models only the classes car, bus and truck are kept, which together correspond to UA-DETRAC's single *vehicle* label. Detections are associated by ByteTrack [17] with the default Ultralytics settings (score thresholds 0.25/0.10, new-track threshold 0.25, track buffer 30 frames, matching threshold 0.8). Track identities feed a virtual-line counter, a speed proxy and an overlay renderer. Masks are drawn from the polygon output in original-image coordinates; the legacy renderer instead resized the whole low-resolution, letter-boxed mask into each box.

Line counting. Let $s_i(t)$ be the coordinate of the box centre of track i in frame t along the axis perpendicular to a virtual line at position ℓ. A track is counted once when its centre changes side of the line; tracks shorter than $L_{min} = 5$ frames are ignored:

$$N = \sum_{i:L_i \geq L_{\min}} \mathbb{1}[\exists t: \sigma_i(t-1)\sigma_i(t) < 0] \quad (1)$$

where $\sigma_i(t) = s_i(t) - \ell$, L_i is the number of frames in which track i is observed and $\mathbb{1}[\cdot]$ is the indicator function. On UA-DETRAC, ℓ is placed perpendicular to the dominant motion direction of the ground-truth trajectories at their median coordinate, and the reference count is Eq. (1) applied to the ground-truth tracks. The line placement therefore uses ground truth; it is the same for every detector and tracker, so comparisons between them are fair, but absolute errors at a hand-placed line may differ.

Speed proxy. With f_v the frame rate of the video and k the window length in frames, the proxy uses video timestamps, not processing time:

$$v_i(t) = f_v \frac{\|c_i(t) - c_i(t-k)\|}{k} \ \text{[px/s]} \quad (2)$$

Without calibration Eq. (2) is a relative motion indicator in pixels per second, not a physical speed, because perspective makes the metres-per-pixel scale vary across the image [8, 9]. The legacy estimator divided pixel displacement by an assumed 8 px/m and by the *wall-clock* interval Δt_{wall} between two processed frames:

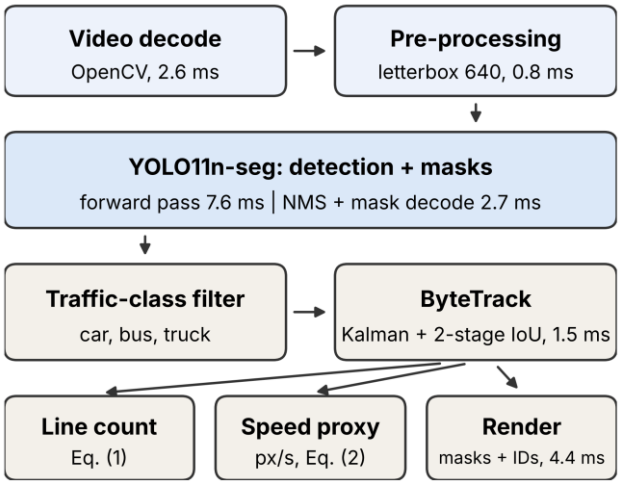


Figure 1: RoadSense pipeline. Stage latencies are frame-weighted means for YOLO11n-seg on an RTX 4060 Ti (FP32) over three UA-DETRAC test sequences and three runs (Section 5.6). The six timed stages sum to 19.6 ms; the measured end-to-end mean is 20.0 ms, the difference being untimed overhead inside the inference call. Line counting and the speed proxy are not timed.

$$v_i^{\text{legacy}} = 3.6\frac{\|\Delta c_i\|/8}{\Delta t_{\text{wall}}} \ [\text{'km/h'}] \quad (3)$$

Because $\Delta t_{wall} \approx 1/F_{proc}$, where F_{proc} is the processing rate, Eq. (3) grows in proportion to how fast the computer runs.

4. DATA AND EXPERIMENTAL PROTOCOL

4.1 Data

COCO val2017 (5,000 images, official annotations) [6] measures accuracy on the five COCO traffic classes (bicycle, car, motorcycle, bus, truck). The pretrained weights were trained on COCO train2017, so val2017 is held out. **UA-DETRAC** [7] provides traffic-camera ground truth. Its 40 official test sequences (56,340 frames, 675,774 vehicle boxes, 2,337 identities) are used for every traffic-camera result. Its 60 training sequences (83,791 frames) are split by sequence into 50 for fine-tuning and 10 for model selection (every sixth sequence in sorted order); no sequence occurs in more than one split and no test sequence is used for training or selection. An integrity audit found no corrupt or missing frames, one frame duplicated across two test sequences and 35 unannotated training frames, which are not used. Because COCO's car, bus and truck all correspond to UA-DETRAC's single *vehicle* label, detection is evaluated class-agnostically. Detections whose centre falls in an annotated ignore region are discarded, following the benchmark protocol. Table 2 summarises the data. UA-DETRAC was obtained from the public Hugging Face mirror abhineet123/ua_detrac (archive SHA-256 checksums recorded); it is used for non-commercial evaluation only and is not redistributed. The three stock video clips used in the earlier version of this work had no recorded source or licence and are excluded.

Table 2. Data used in this study (* every 10th frame)

Set	Split	Seq.	Images	Boxes	Use
COCO val2017	val	–	5,000	–	Accuracy (Tab. 4)
COCO traffic subset	val	–	870	–	Robustness (Fig. 5)
UA-DETRAC	train	50	6,806*	–	Fine-tuning
UA-DETRAC	train	10	1,586*	–	Model selection; Val AP
UA-DETRAC	test	40	56,340	675,774	Detection (every 10th frame), tracking (all frames), latency

4.2 Models and fine-tuning

Pretrained YOLO11n-seg, YOLO11s-seg and YOLO11m-seg weights and the COCO-pretrained YOLO11n detector (Ultralytics release assets) are used unchanged for zero-shot evaluation. The detector of the same size, not the segmentation model, is the zero-shot reference for fine-tuning, so that both sides of the comparison share the same architecture and initial weights. YOLO11n is fine-tuned on the single *vehicle* class with the settings in Table 3; ignore regions are painted grey so that unlabelled vehicles are not learned as background, and the checkpoint with the best validation fitness is kept. The number of epochs was fixed before any result was seen. Three seeds are run with deterministic mode enabled; seed 0 was declared as the detector for tracking before training finished. Software: Python 3.10.11, PyTorch 2.5.1+cu121 (CUDA 12.1), Ultralytics 8.4.142, OpenCV 5.0.0, pycocotools 2.0.11, TrackEval [15]. Hardware: AMD Ryzen 5 7600X 6-Core Processor, 16 GB RAM, NVIDIA GeForce RTX 4060 Ti (8 GB), Windows 11.

Table 3. Fine-tuning configuration (from the run logs and arguments)

Setting	Value
Initial weights	COCO-pretrained YOLO11n (detection)
Epochs / input / batch	12 / 640 px / 32
Optimiser	AdamW, lr 0.002, momentum 0.9 (Ultralytics 'auto'), weight decay 0.0005, 3 warm-up epochs
Precision	Mixed (AMP)
Augmentation	Ultralytics defaults: mosaic (off in the last 10 epochs), HSV jitter, translation 0.1, scale 0.5, horizontal flip 0.5
Seeds; best epoch	0, 1, 2; epochs 12, 9, 12
Training frames	6,806 from 50 sequences (every 10th frame)

4.3 Camera-site audit of the splits

UA-DETRAC sequences are numbered by recording, and consecutive numbers often come from the same fixed camera. Sequence-disjoint splits can therefore still share camera sites. After training, per-sequence median backgrounds (15 frames per sequence) and first frames were inspected visually. A sequence was judged to share a site when road layout, fixed structures and viewing direction matched those of a training sequence. Visual similarity does not prove that two recordings come from the same physical camera, so these labels are judgements rather than ground truth. A first pass compared each validation sequence with its numerical neighbours; a second, independent pass compared every validation sequence without a match against all 50 fine-tuning sequences and every test sequence against all 60 training-partition sequences. By this criterion 7 of the 10 validation sequences share a site with a training sequence: four with near-identical framing, two with slightly shifted framing, and one night-time recording (MVI_39861) of a site that training shows only in daylight, which the first pass had missed. No test sequence was judged to share a site with a training sequence; the 40 test sequences come from 20 sites. Two automatic checks agree but are less sensitive. ORB features with a RANSAC homography on the median backgrounds give at most 30 inliers over all 2,400 test–training pairs (no more than the 99th percentile of all pairs), against 243 and 726 for the two strongest validation–training matches. With a 256-bit difference hash of

every 25th frame, the closest of all test–training frame pairs (2,271 × 3,369) still differs in 67 bits, whereas shared-site validation frames come within 9–31 bits of their training counterparts; neither check detects a day–night pair, so the visual judgement is the operative one. Its evidence images are archived.

4.4 Metrics and statistics

Detection. COCO-style AP averages 101-point interpolated precision over IoU thresholds 0.50:0.05:0.95. AP_{50} and AP_{70} are also reported, AP_{70} being the UA-DETRAC reference threshold [7]. COCO numbers are computed with pycocotools (at most 100 detections per image); UA-DETRAC numbers use every 10th test frame (5,640 frames, at most 300 detections per frame) and an evaluator whose output was cross-checked against pycocotools (Section 5.8). Uncertainty comes from a nonparametric bootstrap with B = 1,000 [14], resampling images for COCO and whole sequences for UA-DETRAC, because frames of one sequence are strongly correlated; intervals are percentile intervals. Because these numbers come from our own protocol (every 10th frame, class-agnostic, COCO-style AP), they are not directly comparable with UA-DETRAC leaderboard results obtained with the official toolkit. Model comparisons are paired: every model is scored on the same resamples, whose identity is checked by a stored fingerprint. Differences are computed from unrounded values, so they can differ by 0.1 from differences of the printed numbers. Because sequences recorded at one site are not independent either, the UA-DETRAC intervals were recomputed with a site-level (cluster) bootstrap over the 20 test sites as a robustness check. No multiplicity correction is applied: only the fine-tuned-versus-zero-shot comparison was pre-declared, and Section 5.3 is exploratory.

Robustness. Six degradations at two severities are applied deterministically to every val2017 image that contains a traffic object: low light (gamma 1.6/2.2, gain 0.55/0.35, read noise σ = 3/6), horizontal motion blur (kernel 9/21 px), Gaussian noise (σ = 10/25), JPEG compression (quality 30/10), resolution loss (2× or 4× down- and up-sampling) and haze (transmission 0.70/0.45) [19, 20]. All conditions are scored with pycocotools on the same images, and relative changes use paired image bootstraps.

Tracking and counting. HOTA, DetA, AssA, MOTA, IDF1 and identity switches are computed with TrackEval on all frames of the 40 test sequences against UA-DETRAC identity ground truth [15, 16]; predicted boxes centred in ignore regions are removed first. Counting error is the per-sequence difference between Eq. (1) on predicted tracks and on ground-truth tracks, summarised as mean absolute error (MAE) and signed bias with a sequence bootstrap.

Latency. Three pre-declared test sequences (the first, middle and last in sorted order) are each processed three times at batch size 1 with no other job running. Twenty warm-up frames are discarded, then 600 (GPU) or 150 (CPU) frames are timed with CUDA synchronisation. Throughput is total frames divided by total wall time; mean, median and 95th-percentile per-frame latency are reported separately.

5. RESULTS

5.1 Accuracy on COCO traffic classes

Table 4 reports accuracy against COCO ground truth. YOLO11m-seg exceeds YOLO11n-seg by 13.9 box AP points (paired 95% CI 12.8–15.2). Small objects are the weakest regime (AP_S 15.9 for YOLO11n-seg), and per-class AP ranges from 25.9 (bicycle) to 63.3 (bus). Model size matters most for small objects: YOLO11m-seg nearly doubles small-object AP relative to YOLO11n-seg (31.0 versus 15.9) but adds only 9.2 points of large-object AP. Mask AP is 7.2–10.5 points below box AP, and this gap widens with model size, so the larger models improve box localisation more than mask quality.

Table 4. COCO val2017 accuracy on the five traffic classes (pycocotools, %); CIs from 1,000 image-level bootstrap resamples

Model	Box AP (95% CI)	AP_{50}	AP_S	AP_L	Mask AP (95% CI)
YOLO11n-seg	40.3 (38.4–42.5)	58.5	15.9	65.4	33.1 (31.3–35.2)
YOLO11s-seg	48.7 (46.9–50.9)	68.2	24.5	71.8	39.0 (37.3–41.2)
YOLO11m-seg	54.2 (52.4–56.5)	74.0	31.0	74.6	43.7 (42.1–45.7)

5.2 Traffic-camera accuracy and the effect of fine-tuning

Table 5 reports vehicle box AP on the UA-DETRAC test set. Zero-shot accuracy on traffic cameras follows the same order as on COCO, but the gap between model sizes is smaller: YOLO11m-seg exceeds YOLO11n-seg by 5.4 points (paired 95% CI 4.3–6.5), against 13.9 points on COCO. The detection-only YOLO11n and the segmentation model of the same size are indistinguishable (difference +0.1, CI −0.4 to +0.6).

Table 5. UA-DETRAC vehicle box AP (%; test: 40 sequences, every 10th frame, ignore regions applied, CIs from 1,000 sequence-level bootstrap resamples). FT: fine-tuned; bold row: mean ± s.d. over the three seeds. Val: the 10 validation sequences of the training partition, same protocol

Model	Setting	Test AP (95% CI)	AP_{50}	AP_{70}	Val AP
YOLO11n-seg	zero-shot	56.5 (53.2–60.1)	81.7	72.8	60.2
YOLO11s-seg	zero-shot	59.9 (56.8–63.3)	84.5	77.4	–
YOLO11m-seg	zero-shot	61.9 (58.9–65.2)	85.0	79.5	65.2
YOLO11n det	zero-shot	56.6 (53.5–60.1)	82.1	73.4	61.0
YOLO11n FT	seed 0	56.8 (52.5–60.9)	81.4	71.1	67.2
YOLO11n FT	seed 1	57.0 (52.4–61.2)	81.0	71.6	67.2
YOLO11n FT	seed 2	56.5 (52.0–60.6)	81.4	70.6	67.5
YOLO11n FT	**mean**	**56.8 ± 0.3**	**81.3±0.3**	**71.1±0.5**	**67.3**

By object size (COCO area bins, pycocotools), small vehicles below 32^2 pixels (7% of boxes) are the weak spot of every zero-shot model (AP 13.6–18.4, against 74.7–78.4 for large vehicles), and the m–n gap is largest for medium vehicles (+5.6 points; 67% of boxes). Fine-tuning shifts rather than raises accuracy: +1.6 points on small and +0.6 on medium vehicles, but −1.2 on large ones (means of seeds).

Fine-tuning gives different results on the two held-out sets. On the 10 validation sequences (1,586 frames, same evaluator), AP rises from 61.0 for the zero-shot YOLO11n detector to 67.3 (mean of three seeds). On the test set the paired difference is −0.2 to +0.4 points across seeds and every 95% CI includes zero; every seed remains below zero-shot YOLO11m-seg. What fine-tuning does change is score calibration [38] (Figure 2). At the legacy fixed threshold of 0.50, recall rises from 0.59 to 0.74 and F1 from 0.73 to 0.82; the F1-optimal confidence moves from 0.25 to 0.42, while the best attainable F1 changes little (0.81 versus 0.83). Resampling the 20 test sites instead of the 40 sequences widens all intervals but changes no conclusion: the fine-tuned mean minus its zero-shot baseline has a site-level CI of −2.7 to +2.5 points, and YOLO11m-seg still exceeds YOLO11n-seg (4.2–6.5).

Table 6 collects these operating points for the zero-shot YOLO11n detector and the fine-tuned detector of seed 0 (the models of Figure 2). The threshold-free metrics barely move or even decrease (AP_{50} −0.7 and AP_{70} −2.3 points), whereas every quantity measured at the fixed confidence of 0.50 improves. Fine-tuning therefore ranks detections about as well as before but

assigns higher scores to true vehicles; the lower AP_{70} indicates that its boxes are slightly less tight at strict IoU thresholds. Together with the size breakdown above (+1.6 points for small and −1.2 for large vehicles), this suggests that the short schedule re-weights the detector towards the small, distant vehicles typical of elevated cameras rather than making it generally more accurate.

Table 6. Operating points of the zero-shot and fine-tuned YOLO11n detectors on the UA-DETRAC test set (fine-tuned: seed 0; precision–recall quantities at IoU 0.5)

Quantity	Zero-shot	Fine-tuned	Change
Box AP, IoU 0.50:0.95 (%)	56.6	56.8	+0.2
AP at IoU 0.50 (%)	82.1	81.4	−0.7
AP at IoU 0.70 (%)	73.4	71.1	−2.3
Recall at confidence 0.50	0.59	0.74	+0.15
F1 at confidence 0.50	0.73	0.82	+0.09
F1-optimal confidence	0.25	0.42	+0.17
Best attainable F1	0.81	0.83	+0.02

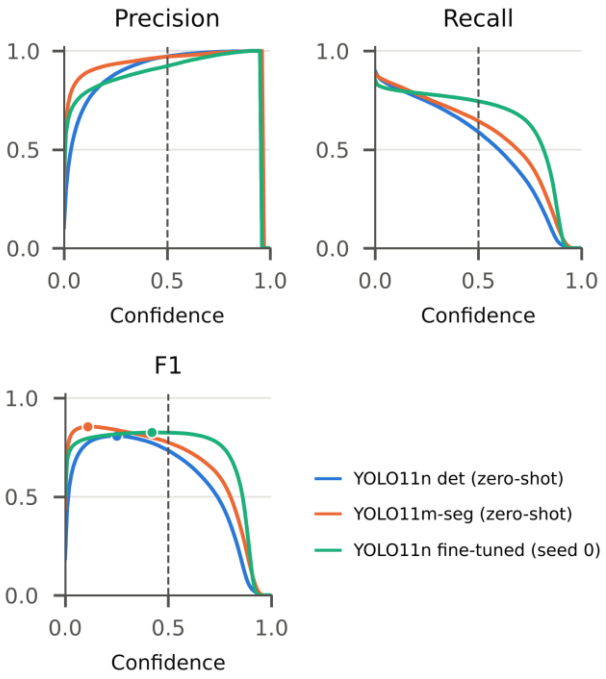


Figure 2: Precision, recall and F1 versus confidence on the UA-DETRAC test set (IoU 0.5) for two zero-shot models and the fine-tuned YOLO11n (seed 0). Dots mark F1 maxima; the dashed line marks the legacy threshold 0.50.

5.3 Why validation and test disagree

Figure 3 breaks the comparison down by sequence. On the validation split, 6 of the 7 sequences from sites that also occur in training improve, and their pooled AP rises from 62.6 to 71.6 (+9.0 points, mean of seeds); the exception is the night-time recording MVI_39861 (−1.0). On the three sequences without a shared site the pooled AP changes by −3.7 to −0.2 points across seeds, although one of them (MVI_63544) gains +9.5; with three sequences this contrast is suggestive rather than conclusive. On the test set the effect is heterogeneous rather than absent: 27 of 40 sequences and 16 of 20 sites improve, with a median site gain of +1.9 points (exploratory sign test over sites $p = 0.01$), but the four losing sites, all recorded at night or on wet roads (judged visually), lose more, down to −12.7 points at a night-time site (MVI_40761–40763) and −15.7 on its worst sequence, so pooled AP does not change. The typical test-site gain is thus several times smaller than the +9.0 points measured at shared validation sites. Model selection on the validation split would have predicted a gain that a new deployment would mostly not see; this is consistent with the leakage pattern described for grouped data [11, 12], here associated with shared camera sites rather than with individual records.

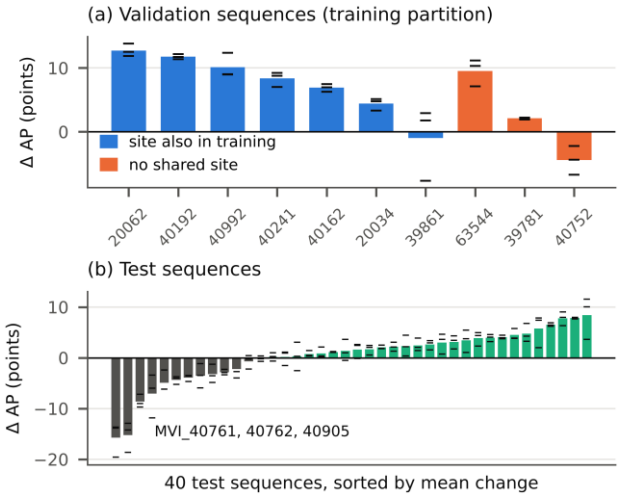


Figure 3: Per-sequence change in box AP (IoU 0.50:0.95, points) from fine-tuning, relative to the zero-shot YOLO11n detector. Bars: mean of three seeds; ticks: individual seeds. (a) Validation sequences, coloured by whether their camera site also occurs in the training sequences (Section 4.3). (b) The 40 test sequences, sorted.

Table 7 condenses these subsets, and Figure 4 contrasts validation and test AP for every model that has a validation score. All zero-shot models lose 3.3–4.4 points between the validation and the test sequences, a gap that reflects the different scenes of the two partitions. For the fine-tuned detector the gap widens to 10.5 points, so about 6–7 points of its validation score are not reproduced at unseen cameras. Model selection by validation AP would have ranked the fine-tuned nano detector above zero-shot YOLO11m-seg (67.3 versus 65.2), whereas on the test cameras it is 5.1 points behind it.

Table 7. Effect of fine-tuning on vehicle box AP (%) by evaluation subset, relative to the zero-shot YOLO11n detector (means of three seeds; * range over seeds)

Subset	Seq. (sites)	Zero-shot	Fine-tuned	Change
Validation, all	10	61.0	67.3	+6.3
Validation, shared site	7	62.6	71.6	+9.0
Validation, no shared site	3	–	–	−3.7 to −0.2*
Test, pooled	40 (20)	56.6	56.8	+0.2
Test, site-level CI of change	40 (20)	–	–	−2.7 to +2.5
Test, median site (16/20 improve)	40 (20)	–	–	+1.9
Test, worst site (night)	3 (1)	–	–	−12.7

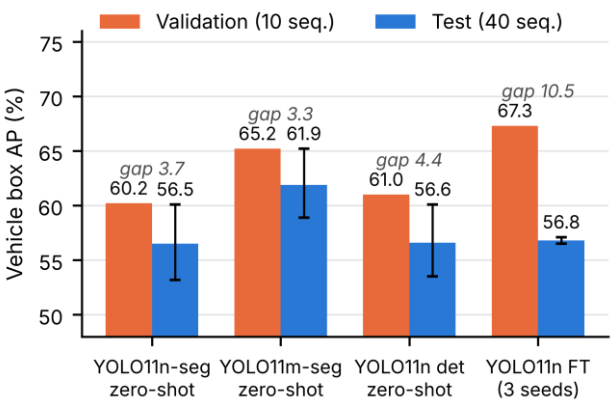


Figure 4: Vehicle box AP on the 10 validation and the 40 test sequences of UA-DETRAC (same evaluator). Error bars: 95% sequence-bootstrap CIs (zero-shot) or ± s.d. over three seeds (fine-tuned). Italic numbers: validation–test gap in points

5.4 Robustness to camera degradations

Figure 5 shows the relative change in box AP on the 870-image COCO traffic subset. For YOLO11n-seg the largest losses are motion blur (S2) (−71%, 95% CI −74 to −67), JPEG compression (S2) (−50%) and motion blur (S1) (−35%). Relative losses are of the same order for all three sizes, but not identical: under motion blur (S2) YOLO11m-seg loses 62% against 71%, and haze barely affects the larger models (−2% at S2 versus −11% for YOLO11n-seg), whereas strong JPEG compression costs YOLO11m-seg slightly more (−56% versus −50%). In absolute terms the m–n gap on this subset narrows from 14.1 points on clean images to 9.1 under motion blur (S2). The claim that the pipeline is robust to lighting, blur and weather is therefore replaced by these measured losses.

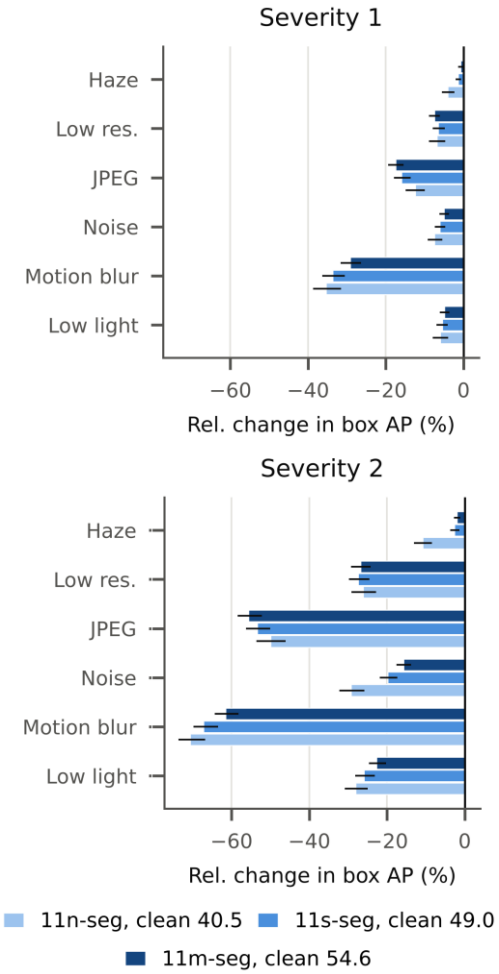


Figure 5: Relative change of traffic-class box AP under 12 deterministic degradations (870 COCO images, pycocotools; legend: clean box AP on this subset); bars: 95% paired bootstrap CIs.

5.5 Tracking and counting against identity ground truth

Table 8. Tracking (TrackEval, %) and counting on the 40 UA-DETRAC test sequences; MAE and bias: line-count error in vehicles per sequence. 11n FT: fine-tuned YOLO11n, seed 0

Det.	Tracker	HOTA	MOTA	IDF1	IDSW	MAE	Bias
11n-seg	Centroid	48.1	53.0	50.7	13,825	8.9	−8.2
11n-seg	ByteTrack	61.1	64.8	72.4	1,816	3.9	−0.1
11n-seg	BoT-SORT	63.1	66.0	73.3	1,392	3.9	−0.2
11m-seg	Centroid	57.5	61.0	63.0	8,307	6.1	−4.7
11m-seg	ByteTrack	66.5	70.9	79.4	684	4.0	−0.7
11m-seg	BoT-SORT	68.6	72.3	79.9	506	4.0	−0.6
11n FT	Centroid	61.0	67.1	67.6	7,290	5.2	−2.1
11n FT	ByteTrack	61.8	62.8	72.8	2,366	5.1	+2.2
11n FT	BoT-SORT	63.6	65.1	73.7	2,192	5.0	+2.0

Table 8 evaluates tracking on all 40 test sequences (1,474 ground-truth line crossings). The original centroid tracker is the weakest association method for every detector: it fragments identities, accumulates the most identity switches and under-counts most strongly. ByteTrack and BoT-SORT differ by at most 2.1 HOTA points, less than the 5.4-point gain from the larger zero-shot detector; the best configuration is zero-shot YOLO11m-seg with BoT-SORT (HOTA 68.6, IDF1 79.9). Consistent with its unchanged AP, the fine-tuned detector tracks about as well as zero-shot YOLO11n-seg with ByteTrack or

BoT-SORT. It does lift the centroid tracker from HOTA 48.1 to 61.0, consistent with its recalibrated scores: that tracker keeps only detections above confidence 0.50, where the fine-tuned model passes more true vehicles. For counting, ByteTrack and BoT-SORT with zero-shot detectors are nearly unbiased (|bias| < 1 vehicle per sequence, MAE 3.9–4.0), whereas the fine-tuned detector over-counts by about 2 vehicles per sequence, together with a higher false-positive count (91,463 versus 60,930 with BoT-SORT).

Figure 6 visualises Table 8. The choice of tracker matters more than the choice of detector: replacing the centroid tracker by BoT-SORT raises HOTA by 15.0 points with YOLO11n-seg and by 11.1 points with YOLO11m-seg, whereas the larger detector adds 5.4–9.4 points for a fixed tracker. BoT-SORT also removes 90% (YOLO11n-seg) and 94% (YOLO11m-seg) of the identity switches of the centroid tracker. The centroid tracker under-counts with every detector, by about 330 of the 1,474 ground-truth crossings in total (about 22%) with YOLO11n-seg, while the modern trackers turn the small negative bias of the zero-shot detectors into a positive one with the fine-tuned detector.

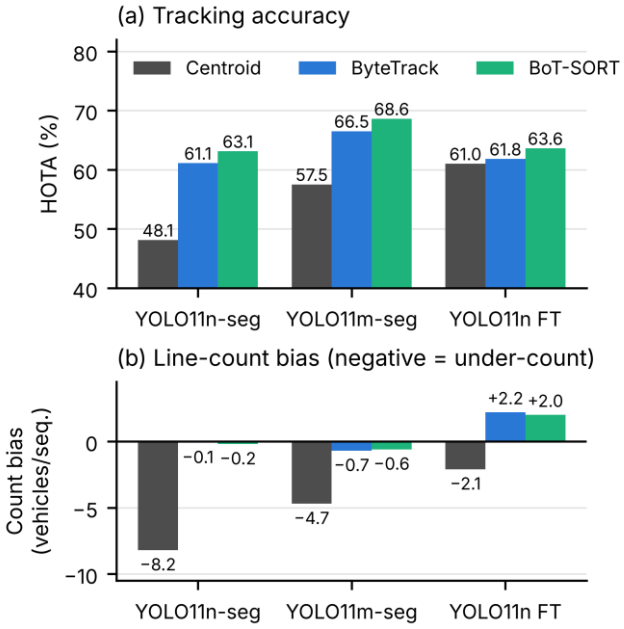


Figure 6: Tracking and counting on the 40 UA-DETRAC test sequences (data of Table 8). (a) HOTA for each detector and tracker. (b) Signed line-count bias in vehicles per sequence; negative values are under-counts.

5.6 Latency budget

Table 9 gives the per-frame budget. With YOLO11n-seg on the GPU the loop sustains 50 frames/s (mean latency 20.0 ms), of which network inference is only 38%. Because fixed costs (decoding, post-processing, rendering) dominate, the larger model costs little on the GPU (49 frames/s for YOLO11m-seg) but a lot on the CPU (21 versus 6 frames/s). On the GPU, run-to-run variation (YOLO11n-seg: 38–59 frames/s across the nine runs) is larger than the differences between model sizes, so the GPU throughputs of the three models should be read as equivalent. Only the GPU configurations exceed the 25 frames/s rate of the UA-DETRAC cameras. A four-panel Matplotlib dashboard redrawn on every frame, reproducing the per-frame redraw of the original design (clearing, plotting, layout and canvas rendering, without its additional video export), takes 99 ms per frame and, because it competes for the CPU, also slows inference and post-processing (to 18 and 17 ms), so the end-to-end mean rises to 149 ms and YOLO11n-seg GPU throughput falls to 7 frames/s; in this reproduction of the original design, per-frame plotting, not the detector, is the real-time bottleneck. Throughput is total frames over total wall time, which is why it is not exactly the reciprocal of mean latency.

Table 9. Per-frame latency (ms) and throughput (frames/s) of YOLO11-seg, FP32, batch 1, three UA-DETRAC sequences × three runs. Dec. JPEG decode, Pre pre-processing, Inf. inference, Post NMS + masks, Trk. ByteTrack, Rnd. rendering; Mean includes untimed overhead

Model	Dev.	Dec.	Pre	Inf.	Post	Trk.	Rnd.	Mean	p95	FPS
11n-seg	GPU	2.6	0.8	7.6	2.7	1.5	4.4	20.0	25.5	50
11n-seg	CPU	1.9	1.1	23.5	14.4	1.9	3.5	46.8	51.6	21
11s-seg	GPU	1.6	0.7	8.1	2.2	1.4	4.2	18.6	21.6	54
11s-seg	CPU	1.9	1.0	51.9	15.4	1.9	3.5	76.1	81.7	13
11m-seg	GPU	1.6	0.7	10.6	1.8	1.4	3.9	20.3	22.6	49
11m-seg	CPU	1.9	1.1	151.9	12.5	1.8	3.2	172.9	177.6	6

The share of network inference in the per-frame budget explains these differences. On the GPU it is 38% for YOLO11n-seg and 52% for YOLO11m-seg; on the CPU it grows from 50% (YOLO11n-seg) through 68% (YOLO11s-seg) to 88% (YOLO11m-seg). Figure 7 combines Tables 5 and 9. On the GPU all three models exceed the camera rate, so YOLO11m-seg gains 5.4 AP points at no practical cost. On the CPU no model reaches 25 frames/s, and only YOLO11n-seg exceeds 20 frames/s; a CPU-only deployment must therefore choose between the fastest model and higher accuracy with frame skipping, for example processing every second frame with YOLO11s-seg, whose effect on tracking was not measured here.

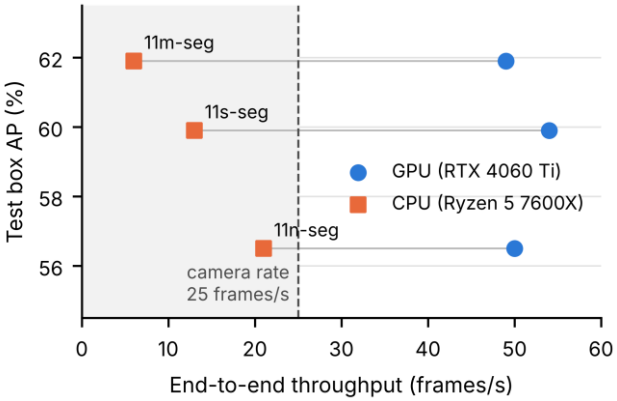


Figure 7: Accuracy–throughput trade-off of the full pipeline: UA-DETRAC test box AP (Table 5) against end-to-end throughput (Table 9). Grey area: below the 25 frames/s rate of the UA-DETRAC cameras.

5.7 Pipeline pitfalls

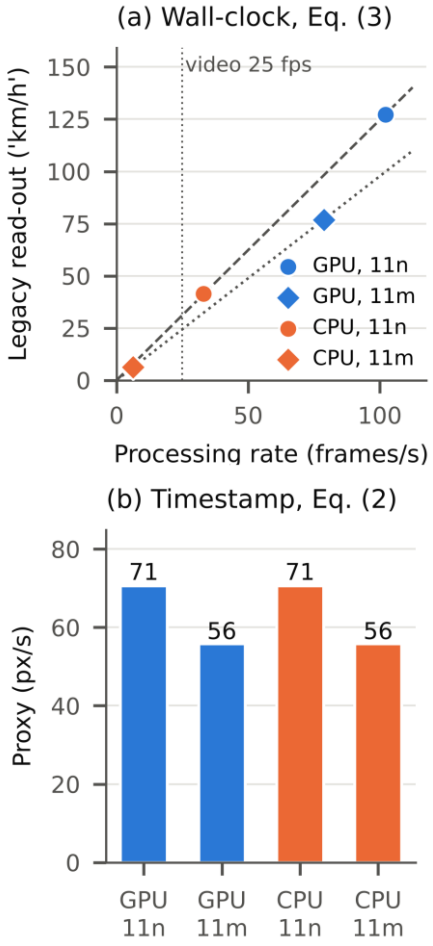


Figure 8: Speed read-outs on the same frames of one UA-DETRAC test sequence under four compute regimes (large markers: mean of three runs; small dots: runs). (a) The legacy wall-clock estimator scales with processing rate (lines through the origin per detector). (b) The timestamp proxy depends only on the detector.

Speed read-out. On 280 frames of a 25 frames/s UA-DETRAC sequence (after 20 warm-up frames, three runs per regime), the legacy estimator, Eq. (3), reports a median of 127 'km/h' with YOLO11n-seg on the GPU (102 frames/s) and 42 'km/h' on the CPU (33 frames/s) for identical motion (Figure 8); with YOLO11m-seg the read-out drops from 77 to 6 'km/h'. For each detector the read-out is proportional to the processing rate within 5%. The timestamp proxy, Eq. (2), gives the same value in every regime for a given detector (71 px/s with YOLO11n-seg, 56 px/s with YOLO11m-seg, which detects a different set of vehicles at confidence 0.50). Without calibration neither is a vehicle speed, so no speed values are reported as physical quantities. The processing rates here cover detection at confidence 0.50 and the centroid tracker only, without rendering, and are not comparable with Table 9.

Mask rendering. Against full-resolution masks from the same model, the legacy overlay reaches a mean IoU of 0.018 (average of six sequences, 2,923 vehicle instances), because the whole letter-boxed mask is shrunk into each box; polygon rendering reaches 0.967 (Figure 9). This measures rendering fidelity, not segmentation accuracy.

Lane drawing before detection. The original code painted Hough lane lines onto each frame before inference. On the UA-DETRAC test frames this changes YOLO11n-seg AP from 56.5 to 56.1 (−0.4 points), a small contamination of the detector input; the revised pipeline draws overlays only after detection.

Table 10 summarises the measured pitfalls. In each case a design choice around the detector, rather than the detector itself, set the limit of the corresponding output: the speed read-out measured the computer rather than the vehicles, the overlay discarded almost all mask geometry, the centroid tracker missed about one vehicle in five at the counting line, and per-frame plotting reduced throughput about seven-fold. Only lane drawing before detection had a small effect.

Table 10. Measured effect of the legacy design choices and of their corrections (UA-DETRAC test data; YOLO11n-seg unless stated)

Component	Legacy	Revised
Speed read-out	6–127 'km/h' on identical frames; proportional to processing rate	71 px/s (11n-seg), 56 px/s (11m-seg) in every regime
Mask overlay (IoU with full-resolution masks)	0.018	0.967
Lane lines drawn before detection (box AP)	56.1	56.5
Per-frame dashboard (mean latency; GPU throughput)	149 ms; 7 frames/s	20.0 ms; 50 frames/s
Association (HOTA; count bias per sequence)	Centroid: 48.1; −8.2	BoT-SORT: 63.1; −0.2
Evaluation artefacts	Self-referential confusion matrix; analytic PR curves	pycocotools and TrackEval against ground truth

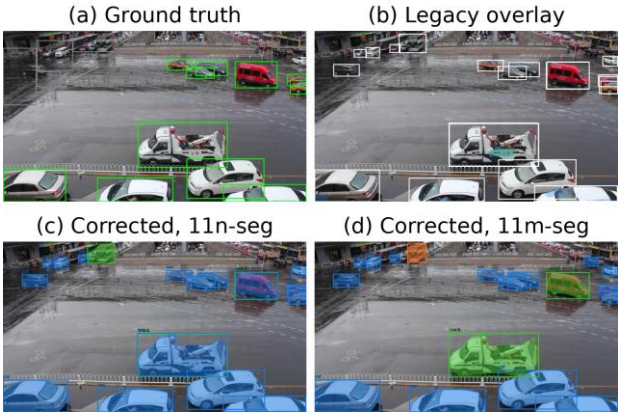


Figure 9: UA-DETRAC test frame at the pre-declared mid-point of a sequence. (a) Ground truth with ignore regions. (b) Legacy overlay: masks collapse to small patches. (c, d) Corrected rendering for YOLO11n-seg and YOLO11m-seg at confidence 0.25. Frames from UA-DETRAC [7], shown for non-commercial research illustration.

5.8 Verification of the reported numbers

Table 11. Verification of reported results

Quantity	Check	Max. difference
UA-DETRAC AP, 7 models	pycocotools from stored detections	$\leq 6 \times 10^{-7}$ AP
UA-DETRAC AP, CIs, paired differences	Independent re-implementation; resamples regenerated from the seed	Identical
COCO AP (box, mask)	pycocotools re-executed on stored predictions	Identical
MOTA, IDF1	Independent CLEAR-MOT / IDF1 implementation	≤ 0.01 points
HOTA	TrackEval re-executed on stored tracks	Identical
Count MAE	Independent counting from stored tracks	≤ 0.08 vehicles
Inference	Re-run on validation set and 1,110 test frames	Bit-identical

Every metric in this paper was recomputed after the experiments (Table 11). Where an independent implementation was possible it was written without reusing the project's metric code; otherwise the original tool was re-executed on the stored artefacts. Inference itself was repeated and reproduced the stored detections exactly on this hardware and software. Small counting differences arise from coordinate rounding in the stored track files and change no conclusion.

5.9 Summary of results

Table 12 answers the three research questions of Section 1 and points to the supporting evidence.

Table 12. Answers to the research questions

Question	Main finding	Evidence
Q1 Zero-shot accuracy	Box AP 40.3–54.2 on COCO traffic classes and 56.5–61.9 on UA-DETRAC vehicles; the n–m gap shrinks from 13.9 to 5.4 points; small vehicles are weakest (AP 13.6–18.4)	Tables 4, 5
Q2 Fine-tuning	Validation AP +6.3 (+9.0 at shared sites); pooled test AP +0.2 (CI includes 0); median test site +1.9, four night or wet-road sites lose up to 12.7; scores recalibrated	Tables 5–7; Figures 2–4
Q2 Tracking and counting	Best: zero-shot YOLO11m-seg + BoT-SORT (HOTA 68.6, count MAE 4.0); centroid tracker under-counts by 8.2 vehicles per sequence	Table 8; Figure 6
Q3 Robustness	Largest losses for YOLO11n-seg: motion blur S2 −71%, JPEG S2 −50%, motion blur S1 −35%	Figure 5
Q3 Pipeline limits	50 frames/s (GPU) and 21 frames/s (CPU) for YOLO11n-seg; per-frame plotting limits it to 7 frames/s; wall-clock speed spans 6–127 'km/h'; legacy mask IoU 0.018	Tables 9, 10; Figures 7–9

6. DISCUSSION AND VALIDITY

What the evidence supports. A training-free deployment is a reasonable baseline on these cameras: zero-shot YOLO11m-seg with a modern tracker gives the best tracking and nearly unbiased line counts. Short fine-tuning on 6,806 frames from other sequences of the same benchmark did not improve pooled accuracy at the test cameras: the typical test site gained about two points, and four night-time or wet-road sites lost much more. It improved accuracy most at camera sites seen during training, and it recalibrated confidence scores, which matters for any component that applies a fixed threshold. Model size helps on traffic cameras, though less than COCO suggests. On a mid-range GPU the full loop runs faster than camera rate for every model size.

Recommendations for practitioners. (i) Split validation data by camera site, not by sequence or frame; otherwise model selection and reported validation gains measure familiarity with the scene [11, 12]. (ii) Report per-site or per-sequence results next to pooled AP, because fine-tuning can help most cameras and still hurt a few badly. (iii) After fine-tuning, re-tune any fixed confidence threshold, or prefer trackers that use low-score detections. (iv) Time the whole loop, including decoding, rendering and plotting, not the network alone. (v) Use video timestamps for motion and calibration for speed.

What it does not support. Metric speed, traffic density and "state-of-the-art" or "highly robust" claims are not supported. The dashboard is an engineering convenience, not a contribution.

Construct and external validity. UA-DETRAC covers fixed elevated cameras in one country and its labels define "vehicle" in a specific way (no two-wheelers); results may not transfer to other camera geometries or vehicle mixes. Synthetic degradations approximate but do not reproduce real night, rain or compression [20]. The counting line is placed from ground-truth trajectories.

Internal and statistical validity. Detection uses every 10th frame, and CIs resample sequences to respect temporal correlation. The three seeds bound optimisation randomness but not the choice of 12 epochs, the optimiser or the model size, which were fixed in advance and not tuned on the test set; longer training, other model sizes or site-disjoint model selection might behave differently. The site-overlap judgement was made by visual inspection after training (two passes; the second found one further shared site), and the per-sequence and per-site analyses in Section 5.3 are exploratory, with only three validation sequences in the no-shared-site group. The site-level test results assume that the 20 test sites are independent. Tracking results use one pre-declared seed. Windows timing varies with background load; repeated runs bound this.

Data licensing and ethics. The original UA-DETRAC website no longer states terms of use; a creator's university page lists the benchmark as a dataset for research use and asks for citation of its first version [7], but states no licence and nothing about reproducing frames. The data are therefore used for non-commercial research evaluation only, are not redistributed, and the frames in Figure 9 are shown solely as research illustration with attribution. The study uses no personal data beyond what is visible in public benchmark video, and no individual is identified or tracked across cameras.

7. CONCLUSION

This study replaced the invalid artefacts of an earlier traffic-monitoring prototype with ground-truth evaluations on COCO and UA-DETRAC and answered three questions. First (Q1), pretrained YOLO11 models transfer reasonably to elevated traffic cameras without any training: they reach 56.5–61.9 vehicle box AP on the 40 UA-DETRAC test sequences, the gap between the nano and medium models is less than half of that on COCO, and small, distant vehicles remain the main weakness. Second (Q2), twelve epochs of sequence-disjoint fine-tuning raised a nano detector's pooled validation AP by 6.3 points, mostly at camera sites seen in training, but not its pooled AP on the test cameras, where small typical gains were cancelled by large losses at night-time and wet-road sites; its main measurable effect was a recalibration of confidence scores. Zero-shot YOLO11m-seg with BoT-SORT gave the best tracking (HOTA 68.6) and nearly unbiased line counts. Third (Q3), the full pipeline runs at 50 frames/s on a consumer GPU and 21 frames/s on a CPU, and the parts that limited the original system were not the detector but the design around it: a wall-clock speed read-out whose values depended on the hardware, a centroid tracker that under-counted, a mask overlay that discarded the mask

geometry, and per-frame plotting that limited throughput to 7 frames/s.

The broader lesson concerns evaluation rather than any particular model. Traffic-camera data are grouped by camera site, and validation that ignores this grouping rewards familiarity with the scene: model selection would have preferred a fine-tuned nano detector that was in fact 5.1 points behind the zero-shot medium model at new cameras. Reporting per-site results, re-tuning confidence thresholds after fine-tuning and timing the whole processing loop are inexpensive steps that change the conclusions a deployment team would draw.

Future scope. Several directions follow from these results. (i) Site-disjoint training and model selection: repeating the fine-tuning study with splits grouped by camera site, longer schedules and larger models will show how much of the remaining test gap fine-tuning can close. (ii) Adverse conditions: the losses at night-time and wet-road sites call for training data or augmentation for low light, glare and rain, and for evaluation on real rather than synthetic degradations. (iii) Adaptation to new cameras: unsupervised or few-shot domain adaptation [37] using the unlabelled video of each new camera could deliver the site-specific gains seen on the validation split without manual annotation. (iv) Calibrated speed and flow: camera calibration or a homography to the road plane, validated against metric ground truth [8, 9], would turn the pixel-based proxy into vehicle speed and enable density and level-of-service estimates. (v) Wider transfer: tests on other benchmarks such as the AI City Challenge data [30] and on mixed traffic with many two- and three-wheelers, as found in South and East Asian cities, together with manual count validation at new sites, will show whether the conclusions hold beyond UA-DETRAC. (vi) Edge deployment: reduced-precision (FP16 or INT8) inference on embedded GPUs and multi-camera processing would make the pipeline usable at intersections without a desktop computer. Releasing the evaluation scripts together with the camera-site labels would allow these extensions to be compared under the same protocol.

7.1 Data and Code Availability

All scripts, configurations, run manifests, raw detections, logs, input checksums and the verification scripts are archived; a canonical results file links every number in this paper to the experiment that produced it. They are available from the authors on request, subject to the dataset terms.

8. REFERENCES

[1] Buch, N., Velastin, S. A., and Orwell, J. 2011. A review of computer vision techniques for the analysis of urban traffic. IEEE Transactions on Intelligent Transportation Systems, 12(3), 920–939. DOI: 10.1109/TITS.2011.2119372.

[2] Sivaraman, S. and Trivedi, M. M. 2013. Looking at vehicles on the road: a survey of vision-based vehicle detection, tracking, and behavior analysis. IEEE Transactions on Intelligent Transportation Systems, 14(4), 1773–1795. DOI: 10.1109/TITS.2013.2266661.

[3] Datondji, S. R. E., Dupuis, Y., Subirats, P., and Vasseur, P. 2016. A survey of vision-based traffic monitoring of road intersections. IEEE Transactions on Intelligent Transportation Systems, 17(10), 2681–2698. DOI: 10.1109/TITS.2016.2530146.

[4] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. 2016. You only look once: unified, real-time object detection. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779–788. DOI: 10.1109/CVPR.2016.91.

[5] Jocher, G. and Qiu, J. 2024. Ultralytics YOLO11, version 11.0.0. Software, AGPL-3.0, https://github.com/ultralytics/ultralytics.

[6] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. 2014. Microsoft COCO: common objects in context. In Proc. European Conference on Computer Vision (ECCV), LNCS 8693, 740–755. DOI: 10.1007/978-3-319-10602-1_48.

[7] Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M.-C., Qi, H., Lim, J., Yang, M.-H., and Lyu, S. 2020. UA-DETRAC: a new benchmark and protocol for multi-object detection and tracking. Computer Vision and Image Understanding, 193, 102907. DOI: 10.1016/j.cviu.2020.102907. Earlier version: arXiv:1511.04136 (2015).

[8] Fernández Llorca, D., Hernández Martínez, A., and García Daza, I. 2021. Vision-based vehicle speed estimation: a survey. IET Intelligent Transport Systems, 15(8), 987–1005. DOI: 10.1049/itr2.12079.

[9] Sochor, J., Juránek, R., Špaňhel, J., Maršík, L., Široký, A., Herout, A., and Zemčík, P. 2019. Comprehensive data set for automatic single camera visual speed measurement. IEEE Transactions on Intelligent Transportation Systems, 20(5), 1633–1643. DOI: 10.1109/TITS.2018.2825609.

[10] Torralba, A. and Efros, A. A. 2011. Unbiased look at dataset bias. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1521–1528. DOI: 10.1109/CVPR.2011.5995347.

[11] Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J. J., Schroeder, B., Thuiller, W., Warton, D. I., Wintle, B. A., Hartig, F., and Dormann, C. F. 2017. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography, 40(8), 913–929. DOI: 10.1111/ecog.02881.

[12] Kapoor, S. and Narayanan, A. 2023. Leakage and the reproducibility crisis in machine-learning-based science. Patterns, 4(9), 100804. DOI: 10.1016/j.patter.2023.100804.

[13] Khanam, R. and Hussain, M. 2024. YOLOv11: an overview of the key architectural enhancements. arXiv:2410.17725.

[14] Efron, B. and Tibshirani, R. J. 1993. An Introduction to the Bootstrap. Chapman & Hall, New York.

[15] Luiten, J., Ošep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., and Leibe, B. 2021. HOTA: a higher order metric for evaluating multi-object tracking. International Journal of Computer Vision, 129(2), 548–578. DOI: 10.1007/s11263-020-01375-2.

[16] Bernardin, K. and Stiefelhagen, R. 2008. Evaluating multiple object tracking performance: the CLEAR MOT metrics. EURASIP Journal on Image and Video Processing, 2008, 246309. DOI: 10.1155/2008/246309.

[17] Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., and Wang, X. 2022. ByteTrack: multi-object tracking by associating every detection box. In Proc. European Conference on Computer Vision (ECCV), LNCS 13682, 1–21. DOI: 10.1007/978-3-031-20047-2_1.

[18] Aharon, N., Orfaig, R., and Bobrovsky, B.-Z. 2022. BoT-SORT: robust associations multi-pedestrian tracking. arXiv:2206.14651.

[19] Hendrycks, D. and Dietterich, T. 2019. Benchmarking neural network robustness to common corruptions and perturbations. In Proc. International Conference on Learning Representations (ICLR). arXiv:1903.12261.

[20] Michaelis, C., Mitzkus, B., Geirhos, R., Rusak, E., Bringmann, O., Ecker, A. S., Bethge, M., and Brendel, W. 2019. Benchmarking robustness in object detection: autonomous driving when winter is coming. arXiv:1907.07484.

[21] Girshick, R. 2015. Fast R-CNN. In Proc. IEEE International Conference on Computer Vision (ICCV), 1440–1448. DOI: 10.1109/ICCV.2015.169.

[22] Ren, S., He, K., Girshick, R., and Sun, J. 2015. Faster R-CNN: towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems 28 (NIPS 2015).

[23] He, K., Gkioxari, G., Dollár, P., and Girshick, R. 2017. Mask R-CNN. In Proc. IEEE International Conference on Computer Vision (ICCV), 2980–2988. DOI: 10.1109/ICCV.2017.322.

[24] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. 2016. SSD: single shot multibox detector. In Proc. European Conference on Computer Vision (ECCV), LNCS 9905, 21–37. DOI: 10.1007/978-3-319-46448-0_2.

[25] Redmon, J. and Farhadi, A. 2017. YOLO9000: better, faster, stronger. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6517–6525. DOI: 10.1109/CVPR.2017.690.

[26] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. 2020. YOLOv4: optimal speed and accuracy of object detection. arXiv:2004.10934.

[27] Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. 2023. YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 7464–7475. DOI: 10.1109/CVPR52729.2023.00721.

[28] Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. 2019. YOLACT: real-time instance segmentation. In Proc. IEEE/CVF International Conference on Computer Vision (ICCV), 9156–9165. DOI: 10.1109/ICCV.2019.00925.

[29] Jocher, G., Chaurasia, A., and Qiu, J. 2023. Ultralytics YOLOv8, version 8.0.0. Software, https://github.com/ultralytics/ultralytics.

[30] Naphade, M., Wang, S., Anastasiu, D. C., Tang, Z., Chang, M.-C., Yang, X., Yao, Y., Zheng, L., Chakraborty, P., Lopez, C. E., Sharma, A., Feng, Q., Ablavsky, V., and Sclaroff, S. 2021. The 5th AI City Challenge. In Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 4258–4268. DOI: 10.1109/CVPRW53098.2021.00482.

[31] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. 2016. Simple online and realtime tracking. In Proc. IEEE International Conference on Image Processing (ICIP), 3464–3468. DOI: 10.1109/ICIP.2016.7533003.

[32] Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. Journal of Basic Engineering, 82(1), 35–45. DOI: 10.1115/1.3662552.

[33] Wojke, N., Bewley, A., and Paulus, D. 2017. Simple online and realtime tracking with a deep association metric. In Proc. IEEE International Conference on Image Processing (ICIP), 3645–3649. DOI: 10.1109/ICIP.2017.8296962.

[34] Majin, J. J., Valencia, Y. M., Stivanello, M. E., Stemmer, M. R., and Salazar, J. D. 2021. A novel deep learning based method for detection and counting of vehicles in urban traffic surveillance systems. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLIII-B2-2021, 793–800. DOI: 10.5194/isprs-archives-XLIII-B2-2021-793-2021.

[35] Ugalino Jr., M. G., Florin, A. F. P., Zamora, M. B. A. I., Aporto, L. J. A., Garcia, L. M., Maralit, P. F. R., Aguilan, K. E. M., Casisirano, J. D. D., Tablang, K. M. C., Pamittan, F. J. O., Fargas Jr., D. C., and Sarmiento, C. J. S. 2026. Urban mobility insights from CCTV: a deep learning approach to traffic flow monitoring. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLVIII-5/W4-2025, 237–244. DOI: 10.5194/isprs-archives-XLVIII-5-W4-2025-237-2026.

[36] Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. 2019. Do ImageNet classifiers generalize to ImageNet? In Proc. 36th International Conference on Machine Learning (ICML), PMLR 97, 5389–5400.

[37] Chen, Y., Li, W., Sakaridis, C., Dai, D., and Van Gool, L. 2018. Domain adaptive Faster R-CNN for object detection in the wild. In Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 3339–3348. DOI: 10.1109/CVPR.2018.00352.

[38] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. 2017. On calibration of modern neural networks. In Proc. 34th International Conference on Machine Learning (ICML), PMLR 70, 1321–1330.