Traffic Monitoring in Low-Visibility Conditions: Dataset, Adapted Detector and Formal Virtual-Line Counting

Vladislav Anisimov
Saint Petersburg Electrotechnical University "LETI"
St. Petersburg, Russia
vyuanisimov@stud.eltech.ru, anisimovvlad.2002@yandex.ru

Oleg Pereziabov, Ilya Afanasyev
Innopolis University
Innopolis, Russia
pereziabov.oa@gmail.com, i.afanasyev@innopolis.ru

***Abstract*—Camera-based traffic monitoring degrades sharply at night and in fog, rain, or dense traffic. This paper presents a complete monitoring pipeline for such conditions and makes four contributions. First, we assemble and publish a low-visibility vehicle dataset of 43,403 annotated images across five classes (car, truck, van, bus, and motorcycle). It harmonizes eight public sources and splits them by camera location to ensure independent training and validation viewpoints. Second, we adapt a medium-scale YOLO11 detector to this dataset and open-source the counting and evaluation code. We provide a precise, closed-form specification of the optimized training objective, which applied work usually describes only informally. Third, we formalize the counting stage with a robust, geometry-based line-crossing rule. This rule is independent of frame rate, survives short detector dropouts, and operates with constant computational cost. Fourth, we report count-level metrics on four real videos with 134 ground-truth vehicles. The adapted model raises the overall count recall from 26.1 to 70.1 per cent (95 per cent confidence interval 61.9 to 77.2), while raising mAP50 from 0.377 to 0.477 on a night test set.**

***Index Terms*—intelligent transportation systems, vehicle detection, object tracking, virtual line counting, nighttime, adverse weather, class imbalance, evaluation protocol**

I. Introduction

Urbanization has amplified the complexity of traffic management, intensifying concerns regarding congestion, public safety, and the efficiency of transport infrastructure. In response to the dual challenges of enhancing mobility and reducing accident rates, intelligent transportation systems have emerged as a primary focus for both the research community and urban planners. Conventional monitoring frameworks reliant on manual observation or primitive inductive loop sensors perform poorly under adverse weather conditions and high traffic density. Consequently, deep-learning-based object detection has established itself as the standard paradigm for real-time traffic monitoring, vehicle classification, and flow analysis.

Among the available architectures, the YOLO (You Only Look Once) framework is prominent, with its YOLO11 generation [1] balancing rapid inference with competitive accuracy. Despite these advancements, adapting such detectors to adverse environments characterized by nighttime capture, precipitation, fog, oblique camera angles, and dense flows remains an unresolved challenge.

A separate methodological challenge lies in how these systems are documented. Applied literature on camera-based vehicle counting frequently details the underlying detector while reducing the counting mechanism to a single sentence, typically described as the intersection between a trajectory and a predefined virtual line. Yet, this counting rule dictates whether a vehicle is registered once, multiple times, or omitted entirely, though its mathematical formulation is seldom provided. Consequently, reported counts cannot be reproduced from the text alone, making it impossible to distinguish the failure modes of the counting stage from those of the detector itself.

This work addresses both gaps. We stress at the outset what is new and what is not. The detector, its loss, the tracker, and the segment-intersection test are all established components. The core contribution of this paper lies instead in the dataset, in the exact specification of every component to ensure reproducibility, and in the count-level evaluation. Specifically:

- A harmonized low-visibility traffic dataset of 43,403 images and five vehicle classes, assembled from eight public sources with a documented class mapping, label sanity checks, and a split by camera location. The dataset and the complete evaluation codebase are published openly [2], [3].
- A precise specification of the adapted detector. A medium-scale YOLO11 model is fine-tuned on the dataset, while the loss, the label assignment, and the treatment of class frequencies are stated in closed form as part of the formal objective description. Although this outlines the objective actually optimized rather than introducing a new one, this documentation is included because applied descriptions are frequently inaccurate.
- A formal specification and open-source implementation of the counting stage. The rule that turns trajectories into counts is written out as a two-sided orientation test on centroid trajectories, featuring a per-track latch, direction attribution, and route counters, alongside its robustness,

cost properties, and failure modes (Algorithm 1). While the test itself is textbook computational geometry, its formalization and released implementation ensure the counting stage becomes strictly reproducible and its errors attributable.
- A count-level evaluation. Image-level metrics are complemented by a count recall computed over four real videos, which is the quantity such a system is ultimately judged on.

The rest of this paper is organized as follows. Section II reviews detector families and traffic datasets. Section III describes the assembly and harmonization of the dataset. Section IV presents the training objective, the optimization strategy, the deployed application and the formal counting model. Section V describes the test material and reports image-level and count-level results. Section VI states the limitations, the reproducibility measures and the ethical considerations, and Section VII concludes.

II. Related Work

A. Evolution of YOLO Detectors for Vehicle Detection

The YOLO architecture remains a benchmark for real-time object detection due to its balance between inference speed and accuracy. Subsequent generations focused on improving detection resilience in challenging environments. For instance, YOLOv10 [4] eliminated non-maximum suppression through consistent dual assignments, reducing computational overhead during continuous monitoring without degrading quality. The YOLO11 generation introduced optimized cross-stage partial structures and advanced spatial attention mechanisms to improve feature aggregation. These structural refinements enhance the detection of small or partially occluded objects under adverse weather and lighting conditions while maintaining inference speed [5]. A parallel line of literature traces these architectural advancements. Diwan et al. [6] document this evolution from the foundational designs through earlier iterations. Similarly, Hidayatullah et al. [7] analyze the transition to recent models. They attribute performance gains to reduced parameter counts in the feature extraction layers and to self-attention modules, which supply the global context required for low-visibility scenes.

Research addressing low-visibility and adverse weather conditions generally follows three distinct paradigms. The first approach fuses information from multiple sensors or network levels, as demonstrated by Zhu et al. [8] in traffic surveillance. The second paradigm adapts the detector to nocturnal conditions through targeted fine-tuning [9]. The third direction exploits visual cues that survive darkness, such as the headlight signatures utilized by Sooksatra et al. [10]. Complementing these approaches, Mudavath and Mamidi [13] analyze specific environmental degradation factors, noting that rain, fog, and snow introduce severe contrast reduction and noise. Consequently, they emphasize the necessity of adapting models to local climatic variations. While integrating complementary hardware like LiDAR improves accuracy under poor visibility [14], it increases system complexity. Ultimately, general surveys identify occlusion handling, scale variation, and domain generalization as the primary persistent challenges in the field [15].

However, robust detection alone is insufficient for traffic monitoring. Tracking-by-detection frameworks must be deployed to associate bounding boxes across consecutive frames. ByteTrack [11] is frequently paired with YOLO detectors because its secondary association step preserves low-confidence boxes, which are highly prevalent in night and fog scenarios. This work utilizes BoT-SORT [12], which extends this multi-stage association by incorporating camera-motion compensation and appearance cues. Crucially, existing literature routinely leaves the subsequent step implicit, omitting the exact mathematical rule that converts a trajectory into a counted event. To address this limitation, the formal geometry of this counting mechanism is detailed as part of the proposed methodology.

B. Datasets for Traffic Analysis

Dataset diversity governs the generalization capability of a detector. UA-DETRAC [16] remains a widely used benchmark for varied weather scenarios. Similarly, the JRDR collection [17] supplies paired rainy and clean images to train weather-resistant models. Despite these efforts, the coverage of extreme environmental conditions remains limited. Patel et al. [18] observe that even major datasets like ApolloScape [19] and IDD [20] contain little adverse weather. The ACDC dataset [21] addresses this gap by providing diverse weather scenarios as a robustness benchmark. However, combined weather effects, such as concurrent fog and rain, still cause the highest rate of detection failures [22]. To establish comparable performance metrics, Zaidi et al. [23] emphasize the critical role of standardized benchmarks like COCO [24], KITTI [25], and BDD100K [26].

The dataset assembled in this work [2] complements these existing resources. It focuses specifically on dense urban traffic, low illumination, and active precipitation. The annotations cover five distinct vehicle classes, including cars, trucks, vans, buses, and motorcycles. To facilitate independent verification and future benchmarking, the dataset is published openly alongside the complete evaluation codebase, including the class harmonization scripts and the formal counting module [3]. A comprehensive analysis of its composition and the underlying curation pipeline is provided in the subsequent dataset description.

III. Dataset Preparation

A. Dataset Architecture and Source Integration

The dataset incorporates material from eight distinct public repositories, as listed in Table I and illustrated in Fig. 1. These sources were selected based on their specific relevance to traffic monitoring rather than raw volume. A heterogeneous collection of traffic imagery from regional road networks [27] provides substantial diversity in vehicle classes, perspective

angles, and illumination levels. During the integration process, vehicle types unique to local transport systems were removed, and the remaining instances were aligned with standard classifications. High-density highway imagery [29] and an additional multi-perspective collection [30] further expand the geographic and structural variety of the data. Similarly, static wide-angle camera sequences captured during precipitation [28] provide coverage for nearly all target classes. To ensure rigorous evaluation, data from one camera installation were reserved exclusively for validation, while the remaining locations were allocated to the training partition.

Specific components were incorporated to reinforce classes that were under-represented in the initial selections. The UA-DETRAC benchmark [16] contributes 10,000 images capturing highways and intersections under diverse weather conditions and temporal intervals. An additional 10,000 intersection instances [31] introduce crucial chromatic and spatial variations. High-density traffic flows at a single junction [32] provide detailed close-range interactions, while a focused set of 5,000 low-light instances [33] directly addresses detection performance during twilight and nocturnal periods.

The documented source capacities in Table I represent the baseline metrics reported in their respective original releases. Consequently, these individual values do not sum directly to the final volume of the unified dataset. Several source repositories omit explicit instance counts entirely. Furthermore, the subsequent standardization process excludes any image that contains no valid targets once non-target classifications are removed. The final dataset consists of the remaining 43,403 verified images.

The validation and training partitions are strictly isolated to prevent data leakage. Repositories utilized across both subsets were divided based on distinct viewpoints or geographic locations rather than through random assignment. This design ensures that the training and validation sets share no identical camera perspectives. This structural division is embedded directly within the public release, allowing external verification of the partition integrity and enabling independent duplication audits as detailed in the closing discussion.

B. Class Harmonization and Imbalance

The selected sources originally employed distinct classification systems. To ensure consistency, all annotations were mapped to a unified target taxonomy consisting of five vehicle classes: car, truck, van, bus, and motorcycle. Fine-grained morphological variants, such as sedans, hatchbacks, and sport utility vehicles, were consolidated into the car class, while minibuses were merged into the bus category. Non-target classifications outside the scope of this study, including pedestrians, bicycles, emergency vehicles, and three-wheeled transports, were excluded. Image files left without any valid annotations following this exclusion process were systematically removed from the collection to prevent the accidental inclusion of empty background samples during training.

Six of the eight repositories were retrieved from community-driven platforms (Kaggle or Roboflow Universe) where annotation quality is not guaranteed, necessitating rigorous verification procedures. The taxonomic mapping for each source was finalized only after a systematic visual inspection of image samples with overlaid source labels. This process allowed the identification of unique local transport types and special vehicle classes requiring explicit handling, while simultaneously serving as a quality control check for bounding box alignment.

During the conversion to the unified format, malformed records were systematically discarded. This filtering excluded

TABLE I. CONSTITUENT SOURCES OF THE ASSEMBLED DATASET

Set	Content and conditions	Ref.
1	Mixed Indian roads, varied angles, lighting and weather	[27]
2	Four static fisheye cameras in rain, one location held out for validation	[28]
3	Busy highways, all target classes, limited diversity	[29]
4	Additional viewpoints, largest single contribution	[30]
5	UA-DETRAC, 10,000 images, highways and intersections	[16]
6	Intersections, 10,000 images, color and spatial variety	[31]
7	Dense flow at one intersection, small volume	[32]
8	Dusk and night, 5,000 images, low illumination	[33]

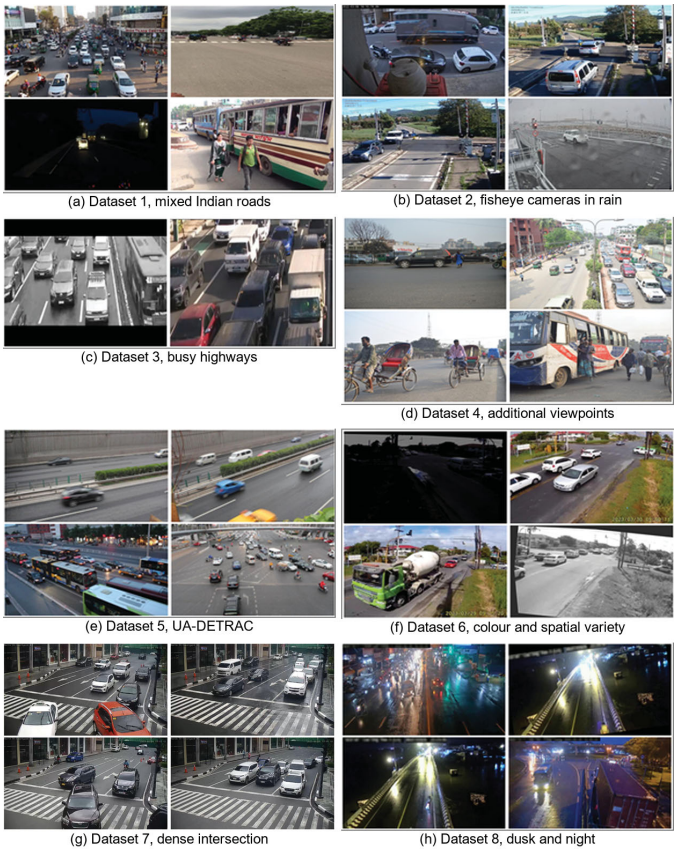


Fig. 1. One sample per constituent source of the assembled dataset. The sources differ in camera geometry, illumination, weather and traffic density

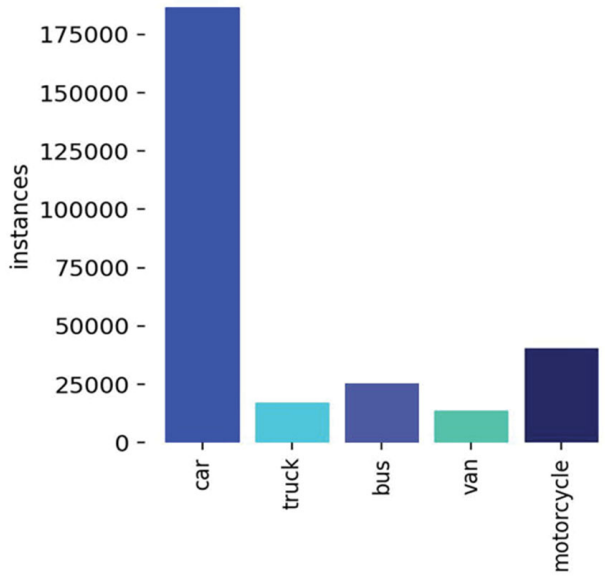


Fig. 2. Class distribution of the assembled dataset, which reflects the real-world ratio of vehicle types on urban roads

corrupted annotation lines, class identifiers outside the documented source vocabulary, and bounding boxes lacking a positive spatial area. Repositories exhibiting severe visual inconsistencies during inspection were rejected entirely, leaving the eight verified sources listed in Table I. Because the original annotators remain anonymous, formal inter-annotator agreement metrics could not be established. Consequently, these verification steps were designed to mitigate gross structural errors rather than to quantify pixel-level label precision. This residual annotation noise partially explains why the validation mean average precision at strict overlap thresholds remains noticeably lower than the metrics obtained at standard thresholds, as analyzed in the subsequent empirical evaluation.

Let $n_{c,d}$ be the number of annotated instances of class $c \in \mathcal{C}$ contributed by source $d \in \{1, \dots, 8\}$ after mapping. The class count and the class frequency of the assembled dataset are

$$n_c = \sum_{d=1}^{8} n_{c,d}, \qquad f_c = \frac{n_c}{\sum_{c' \in \mathcal{C}} n_{c'}}. \tag{1}$$

Sources 5 and 6 were added specifically to raise n_c for truck, van and bus, while keeping the frequencies f_c of (1) close to the real-world traffic mix, in which passenger cars form roughly 70 to 80 per cent of urban traffic [34], and in which cars outnumber trucks by about twelve to one. The resulting distribution is shown in Fig. 2. The imbalance was therefore addressed at the level of the data, by choosing sources, and not at the level of the loss: no per-class re-weighting was applied in the experiments reported here. This is a deliberate choice, since a detector deployed for traffic statistics should reproduce the operational class mix rather than a balanced one.

IV. Methodology

The methodology has two parts: adapting the detector, and the video application that turns detections into traffic statistics. The medium-scale variant YOLO11-m, with 20.1 million parameters and 68.2 GFLOPs at 640 pixels, was selected as the backbone because it balances computational cost against accuracy. Published comparisons place it above earlier YOLO generations on COCO [35] and competitive with the transformer-based RT-DETRv2 [36] at a lower computational cost [37]. That trade-off matters for a system that must keep up with a live camera.

A. Training Objective

Equations (2) to (6) define the exact objective minimized during fine-tuning. While standard in the Ultralytics framework, stating them explicitly prevents ambiguity: (2) assigns ground-truth objects to grid cells, (4) classifies them via soft targets, (5) regresses the bounding boxes using complete IoU, and (6) refines the box boundaries via distribution focal loss.

The objective stated here is the default detection loss of the framework used [1]; we write it out because it is the loss actually optimized in the experiments and because applied descriptions of it are frequently inaccurate, not because it is modified. YOLO11 is anchor-free. For every grid cell i of three strides it predicts a class-score vector $\hat{\mathbf{p}}_i \in [0,1]^{|\mathcal{C}|}$ and, for each box side $e \in \{l, t, r, b\}$, a discrete distribution S_i^e over $r_{\max}+1 = 16$ distance bins. Ground-truth objects are matched to cells by a task-aligned assigner [39]: each object g selects the ten highest-scoring cells under the alignment metric

$$m_{i,g} = \hat{p}_{i,c_g}^{\,\alpha} \, \mathrm{IoU}(\hat{\mathbf{b}}_i, \mathbf{b}_g)^{\beta}, \qquad \alpha = 0.5, \ \beta = 6, \tag{2}$$

where IoU denotes intersection over union. The classification target of a selected cell is a soft label $t_{i,c_g} \in [0,1]$, obtained by rescaling $m_{i,g}$ of (2) so that its maximum over the selected cells equals the largest intersection over union achieved for that object; all remaining targets are zero. With $\mathcal{P}$ the set of assigned cells, $w_i = \sum_c t_{i,c}$ and the normalizer $Z = \max(\sum_{i,c} t_{i,c}, 1)$, the objective is

$$\mathcal{L} = \lambda_{\text{box}} \mathcal{L}_{\text{CIoU}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} + \lambda_{\text{dfl}} \mathcal{L}_{\text{DFL}}, \tag{3}$$

with the fixed gains $(\lambda_{\text{box}}, \lambda_{\text{cls}}, \lambda_{\text{dfl}}) = (7.5,\ 0.5,\ 1.5)$ and the three terms

$$\mathcal{L}_{\text{cls}} = \frac{1}{Z} \sum_i \sum_{c \in \mathcal{C}} \Big[-t_{i,c} \log \hat{p}_{i,c} - (1 - t_{i,c}) \log(1 - \hat{p}_{i,c}) \Big], \tag{4}$$

$$\mathcal{L}_{\text{CIoU}} = \frac{1}{Z} \sum_{i \in \mathcal{P}} w_i \Big[1 - \mathrm{IoU}_i + \frac{\rho_i^2}{d_i^2} + \alpha_i v_i \Big], \tag{5}$$

$$\mathcal{L}_{\text{DFL}} = -\frac{1}{Z} \sum_{i \in \mathcal{P}} \frac{w_i}{4} \sum_e \Big[\bar{y}_i^e \log S_i^e(y_{i-}^e) + \underline{y}_i^e \log S_i^e(y_{i+}^e) \Big], \tag{6}$$

where ρ_i is the distance between the centers of the predicted and the ground-truth box and d_i the diagonal of the smallest box enclosing both. In (5), $w_{g(i)}$ and $h_{g(i)}$ denote the width and height of the ground-truth box assigned to cell i, while

$\hat{w}_i$ and $\hat{h}_i$ are the predicted dimensions; the term $v_i = \frac{4}{\pi^2}(\arctan\frac{w_{g(i)}}{h_{g(i)}} - \arctan\frac{\hat{w}_i}{\hat{h}_i})^2$ and $\alpha_i = v_i/(1 - \mathrm{IoU}_i + v_i)$ carry the aspect-ratio consistency of the complete intersection over union [40]. In (6), y_i^e is the ground-truth distance from the cell center to side e in stride units, $y_{i-}^e = \lfloor y_i^e \rfloor$ and $y_{i+}^e = y_{i-}^e + 1$ are the neighboring bins, $\bar{y}_i^e = y_{i+}^e - y_i^e$ and $\underline{y}_i^e = y_i^e - y_{i-}^e$ are the interpolation weights, so that the term is the two-bin cross-entropy of the generalized focal loss [41].

Three aspects of this objective require clarification due to frequent inaccuracies in literature. First, the regression term (5) is a complete intersection-over-union loss, not a traditional coordinate regression. Second, the classification term (4) computes per-class binary cross-entropy against soft targets; this replaces softmax cross-entropy against one-hot labels, enabling the integration of the alignment metric (2). Finally, the weights in (3) are fixed hyperparameters rather than learned parameters, and the detection model excludes any segmentation terms.

B. Optimization Strategy

The experiment tracker [38] recorded the recipe of the adaptation run, and it is worth stating exactly, because the training script fixes only the dataset and the epoch budget: every remaining value is a default of the release used, Ultralytics 8.3.86. Optimizer selection was left to the framework, which chose stochastic gradient descent with an initial learning rate of 10^{-2} and a momentum of 0.9, and applied a weight decay of $5 \cdot 10^{-4}$ to the convolutional and linear weights only. The learning rate rises over a warm-up of three epochs and then decays linearly towards 10^{-4} at the end of the budget. The batch size was 16 at an input resolution of 640 pixels, which fills the memory of the graphics processor used, and the seed was fixed.

The epoch budget was 70. Two framework mechanisms are tied to that budget and should be stated to avoid ambiguity: the early-stopping patience was left at its default of 100 epochs, so early stopping could not trigger within the budget, and mosaic augmentation was scheduled to switch off for the last ten epochs of the budget, that is from epoch 61. The run did not reach either point. Training was terminated after epoch 45, because the validation mean average precision had not changed measurably for 25 epochs (Section V-B); the checkpoint with the best validation mAP50-95, as selected by the framework, is the model evaluated throughout. Two consequences follow. The mosaic-off phase was never reached, so the evaluated model was trained with mosaic active in every epoch. And at epoch 45 the linear schedule had brought the learning rate only to roughly a third of its initial value, so the reported model is, if anything, slightly under-trained relative to the full recipe; the plateau of Section V-B indicates that the remaining gain would have been small. Augmentation combined mosaic composition, horizontal flipping, random scaling and translation, and jitter in hue, saturation and value, which approximate the variation in lighting, weather and traffic density of the deployment environment. Table II lists the configuration in full, including the transformations that were left switched off, so that the run can be repeated without reading the framework defaults out of a configuration file.

TABLE II. TRAINING CONFIGURATION OF THE ADAPTED DETECTOR, AS RECORDED BY THE EXPERIMENT TRACKER

Setting	Value
Detector	YOLO11-m, 20.1 M parameters, 68.2 GFLOPs
Framework	Ultralytics 8.3.86, PyTorch 2.6.0, CUDA 11.8
Hardware	NVIDIA GeForce RTX 3060 Ti, 8 GB
Input size	640×640
Batch size	16
Optimizer	SGD, selected by the framework
Learning rate	10^{-2}, linear decay towards 10^{-4} over the budget
Warm-up	3 epochs
Momentum	0.9
Weight decay	$5 \cdot 10^{-4}$, applied to weights only
Loss gains	box 7.5, class 0.5, distribution 1.5
Epoch budget	70; run terminated after epoch 45 at the validation plateau
Early stopping	patience 100 (default), not triggered
Checkpoint	best validation mAP50-95, selected by the framework
Seed	0, deterministic kernels
Mosaic	1.0, scheduled off for epochs 61–70 (not reached)
Hue, saturation, value	0.015, 0.7, 0.4
Translation, scale	0.1, 0.5
Horizontal flip	0.5
Random erasing	0.4
Switched off	rotation, shear, perspective, vertical flip, mixup, copy-paste
Epoch time	about 33 min
Detector latency	36.5 ms per frame at 640 px

C. Video Analysis Application

The deployed system consists of two modules written in Python. The first, line marking, is a configuration step run once per camera. The first frame of the video is displayed, the operator marks a line by selecting two points, and then enters its name and its type, entry or exit. Coordinates, names and types are stored as a JSON description of the scene.

The second module performs the analysis. It loads the video together with the scene description, applies the detector to every frame for the five vehicle classes, and passes the detections to the tracker that the framework bundles, BoT-SORT [12] in the release used here, which assigns a persistent identity to each vehicle. The trajectory of every identity is then tested against every virtual line, crossings are recorded, and the resulting events are accumulated into per-route and per-class counters. The complete pipeline is shown in Fig. 3.

D. Formal Model of the Counting Stage

1) Notation: Frames are indexed by $t \in \mathbb{N}$. The detector maps frame I_t to detections $\mathcal{D}_t = \{(\mathbf{b}_i, c_i, s_i)\}$ with box $\mathbf{b}_i$, class $c_i \in \mathcal{C}$ and confidence $s_i \in [0, 1]$. The tracker assigns an identity $k \in \mathcal{K}$ to detections across frames. Let $\mathcal{T}_k \subseteq \mathbb{N}$ be the set of frames in which track k is observed and $\mathbf{c}_t^k$ the centroid of its box at frame $t \in \mathcal{T}_k$. Because detections can be

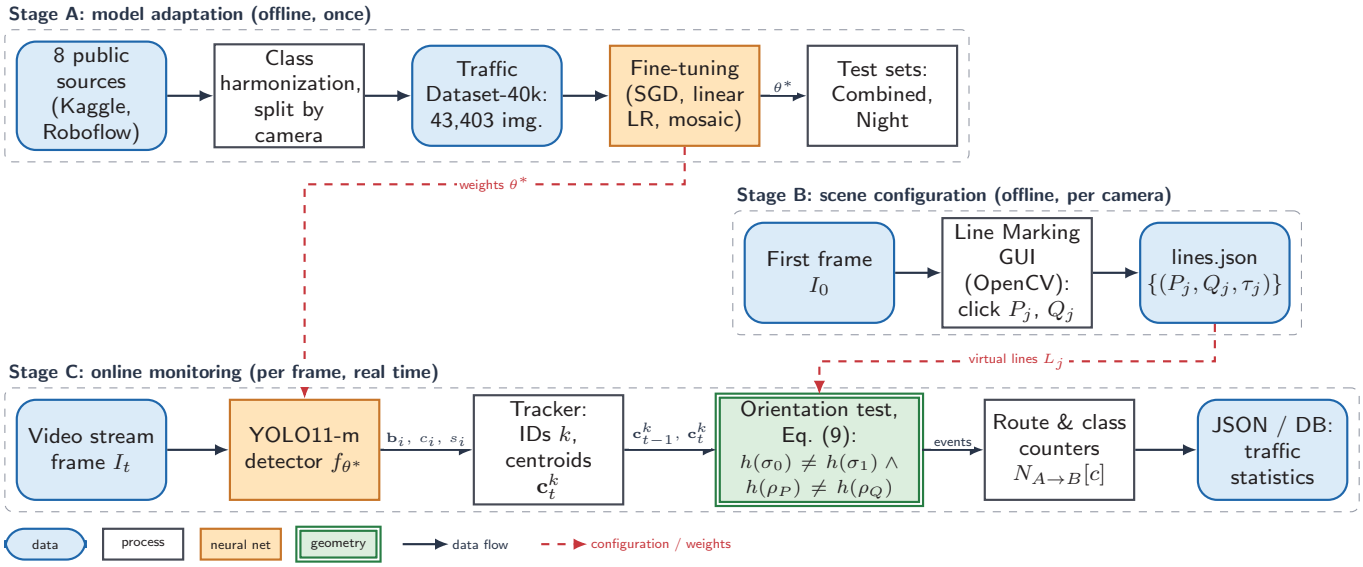


Fig. 3. System pipeline. Stage A adapts the detector once, harmonizing eight public sources to five classes and splitting them by camera. Stage B configures a camera once, marking entry and exit lines on the first frame. Stage C runs per frame, tracking detections, testing consecutive centroids against every line with the two-sided orientation test of (9), $h(\sigma_0) \neq h(\sigma_1) \wedge h(\rho_P) \neq h(\rho_Q)$ with the half-plane indicator h of (8), and updating the route and class counters

missed, $\mathcal{T}_k$ may contain gaps, so we write $t' = \max\{s \in \mathcal{T}_k : s < t\}$ for the previous observation and $\mathbf{e}_t^k = \mathbf{c}_t^k - \mathbf{c}_{t'}^k$ for the displacement between two consecutive observations. A scene is configured with m virtual lines $L_j = (P_j, Q_j, \tau_j)$, where $P_j, Q_j \in \mathbb{R}^2$ are the marked end points, $\mathbf{d}_j = Q_j - P_j \neq \mathbf{0}$, and $\tau_j \in \{\text{entry}, \text{exit}\}$. For $\mathbf{u}, \mathbf{v} \in \mathbb{R}^2$ the planar cross product is the scalar $\mathbf{u} \times \mathbf{v} = u_x v_y - u_y v_x$.

The counting logic relies on three geometric primitives. Equation (7) defines a signed distance to a line, (8) resolves degenerate cases where a centroid lies exactly on the line, and (9) combines them into a robust segment-intersection test that fires exactly once per crossing.

2) Side function: The signed area

$$\sigma_j(\mathbf{p}) = \mathbf{d}_j \times (\mathbf{p} - P_j) \tag{7}$$

is positive on one side of the infinite line through P_j and Q_j, negative on the other and zero on the line itself, and $|\sigma_j(\mathbf{p})|/\|\mathbf{d}_j\|$ is the distance from $\mathbf{p}$ to that line. To make every subsequent predicate total, points on the line are assigned to the non-negative side through the half-plane indicator

$$h(x) = [\, x \geq 0 \,] \in \{0, 1\}, \tag{8}$$

so that $h(\sigma_j(\mathbf{p}))$ is defined for every $\mathbf{p}$, including points exactly on the line.

3) Crossing event: Track k crosses line j at observation t when the chord between its two consecutive centroids and the line segment intersect, that is when

$$\chi_j^k(t) = \big[h(\sigma_j(\mathbf{c}_{t'}^k)) \neq h(\sigma_j(\mathbf{c}_t^k))\big] \wedge \big[h(\rho_t^k(P_j)) \neq h(\rho_t^k(Q_j))\big] \tag{9}$$

holds, with $\rho_t^k(\mathbf{p}) = \mathbf{e}_t^k \times (\mathbf{p} - \mathbf{c}_{t'}^k)$ and σ_j, h from (7) and (8). The first factor states that the two consecutive centroids lie in different half-planes of the virtual line, one of them closed and the other open. The second states that the two end points of the virtual line lie on different sides of the motion chord, which is what confines the crossing to the marked segment instead of its infinite extension. Whenever no quantity is exactly zero, (9) coincides with the classical strict orientation test $\sigma_j(\mathbf{c}_{t'}^k)\sigma_j(\mathbf{c}_t^k) < 0 \wedge \rho_t^k(P_j)\rho_t^k(Q_j) < 0$ for segment intersection [42]. The half-plane convention matters only in the degenerate cases, and there it supplies what the strict test lacks. Under the strict test, a centroid that lands exactly on the line at observation t makes the product zero both at t and at the next observation, so the crossing is never registered; under (9) that centroid belongs to the non-negative side, the sequence of sides along the trajectory is well defined, and the event fires exactly once, either at t or at the next observation, depending on which side the track came from. A chord that runs along the line without leaving its half-plane produces no event, which is correct, since nothing has crossed. No tolerance band ε is needed, because the test compares sides rather than distances. With floating-point centroids an exact zero is a measure-zero event, so on real tracks (9) and the strict test agree on every observed crossing; the convention is what makes the model total and provably correct, and it costs nothing to implement.

4) Direction and crossing instant: The direction of the crossing is

$$\delta_j^k(t) = \operatorname{sgn}\left(\mathbf{d}_j \times \mathbf{e}_t^k\right) \in \{-1, +1\}, \tag{10}$$

which separates the two travel directions across the same line. The route counters defined below do not depend on δ, because in the evaluated scenes every marked line is traversed in a single direction and the entry or exit type τ_j already fixes the sense of travel. The direction is nevertheless computed and stored with every event, at no extra cost since the cross

Algorithm 1 Virtual-line crossing and route counting, one frame

Require: lines $\{(P_j, Q_j, \tau_j)\}_{j=1}^{m}$, tracks observed at t with previous observation t', $h(x) = [x \geq 0]$

for each track k with centroids $\mathbf{c}_{t'}^k$ and $\mathbf{c}_t^k$ **do**
  $\mathbf{e} \leftarrow \mathbf{c}_t^k - \mathbf{c}_{t'}^k$
  for each line j whose latch for k is not set **do**
    $\sigma_0 \leftarrow \mathbf{d}_j \times (\mathbf{c}_{t'}^k - P_j)$, $\sigma_1 \leftarrow \mathbf{d}_j \times (\mathbf{c}_t^k - P_j)$
    $\rho_P \leftarrow \mathbf{e} \times (P_j - \mathbf{c}_{t'}^k)$, $\rho_Q \leftarrow \mathbf{e} \times (Q_j - \mathbf{c}_{t'}^k)$
    if $h(\sigma_0) \neq h(\sigma_1)$ **and** $h(\rho_P) \neq h(\rho_Q)$ **then**
      set the latch, record the event $(k, j, t, \mathrm{sgn}(\mathbf{d}_j \times \mathbf{e}))$
    end if
  end for
  if k has an entry event A, a later exit event B, and is not yet counted **then**
    $N_{A\to B}[\hat{c}_k] \leftarrow N_{A\to B}[\hat{c}_k] + 1$, mark k counted
  end if
end for

product is a by-product of (9), for two uses that arise in deployment: a bidirectional lane can be covered by one line whose two directions feed separate counters, and an event whose δ contradicts the type of the line (an exit line crossed inward, for instance a U-turn or a tracker error that drags a box backwards) can be filtered out before it is admitted to a route. When a time stamp is required, the crossing point is $\mathbf{c}^\star = \mathbf{c}_{t'}^k + u\,\mathbf{e}_t^k$ and the sub-frame instant is $t^\star = t' + u\,(t - t')$, with

$$u = \frac{(P_j - \mathbf{c}_{t'}^k) \times \mathbf{d}_j}{\mathbf{e}_t^k \times \mathbf{d}_j} \in [0, 1]. \tag{11}$$

5) Latching, routes and counters: Each pair of a track and a line fires at most once. The event time is $E_j^k = \min\{t \in \mathcal{T}_k : \chi_j^k(t)\}$, after which a latch suppresses further events for that pair, which removes the double counts that centroid jitter around a line would otherwise produce. The route of a track is its first entry line followed by its first later exit line,

$$a_k = \arg\min_{j:\,\tau_j = \text{entry}} E_j^k, \quad b_k = \arg\min_{j:\,\tau_j = \text{exit},\ E_j^k > E_{a_k}^k} E_j^k, \tag{12}$$

its class $\hat{c}_k$ is the confidence-weighted majority vote over its observations, and the route counters accumulate

$$N_{A\to B}[c] = \big|\{k \in \mathcal{K} :\ (a_k, b_k) = (A, B),\ \hat{c}_k = c\}\big|. \tag{13}$$

Algorithm 1 states the per-frame procedure that realizes (7) to (13).

6) Properties: The model has four properties that matter in deployment. First, gap robustness: (9) tests the chord between consecutive observations, so a crossing is detected for any displacement $\|\mathbf{e}_t^k\|$, including fast vehicles seen at a low frame rate and tracks interpolated through a detector dropout. A proximity test of the form $|\sigma_j|/\|\mathbf{d}_j\| < \varepsilon$, which is the usual shortcut, misses every crossing whose per-frame displacement exceeds 2ε; this is a geometric statement about the two predicates and is not tested empirically here. Second, cost: four cross products per pair of a track and a line, that is $O(|\mathcal{K}_t|\, m)$ per frame, where $\mathcal{K}_t \subseteq \mathcal{K}$ is the set of tracks active at frame t (so $|\mathcal{K}_t|$ is the number of detections in that frame), and m is the number of virtual lines; this cost is negligible next to detector inference. Third, determinism: given the same detections and identities, the counters are reproducible, and the same four products already yield the direction of (10) and the time stamp of (11). Fourth, the failure modes are explicit and attributable to stages. The latch removes exactly one error source, the repeated firing of a single track at a single line, so centroid jitter cannot inflate a count. It does not protect against errors made upstream: an identity switch between the entry and the exit crossing destroys the route of that vehicle and produces an undercount, while a spurious track, whether born from a false-positive detection or from a vehicle whose identity is re-assigned after a switch, can traverse an entry and an exit line and produce a count that corresponds to no vehicle or to a vehicle already counted. The second kind of error requires a wrong track to persist across two lines, which is less likely than the first kind, which requires only a single lost association; the counters are therefore expected to be biased towards undercounting, but a strict lower bound would require event-level matching to ground truth, which Section V does not perform. Section V shows the expected undercounting signature.

V. Results of Experiments

The fine-tuned model is compared against the baseline YOLO11-m model trained on COCO [24], on two held-out image sets and on four videos. Metrics on the image sets are the per-class F1 score at the F1-optimal confidence, together with mAP50-95, mAP50 and mAP75, which summarize precision across intersection over union thresholds.

A. Evaluation Datasets and Video Selections

The combined test dataset consists of daytime and nighttime images of high-density highway traffic captured by static cameras. The night test set represents the nighttime subset of this collection. Crucially, all evaluation cameras are entirely disjoint from those utilized in the training and validation phases. A portion of this evaluation data originates from the same public benchmark as one of the training sources, with the partitioning executed strictly by camera location rather than by dataset source. Both evaluation sets maintain the original four-class vocabulary of car, truck, bus, and motorcycle, where vans are categorized under the bus label. These annotations were used as released following the identical format validation protocols applied to the training data. The systemic implications of this vocabulary mismatch on evaluation metrics are detailed separately below.

The video evaluation suite comprises four distinct segments from static cameras, reflecting the specific environmental scenarios illustrated in Fig. 7: fog, a night intersection, a high-density highway at night, and a complex intersection captured from a long-distance viewpoint. These video sequences are completely independent of the eight image training sources. The reference ground truth was established via exhaustive

manual inspection of each segment, recording the number of vehicles traversing both the entry and exit boundaries for each designated route. In total, the dataset encompasses 134 vehicles across twelve routes. This comprises 31 vehicles on three routes in the fog scenario, 16 on two routes at the night intersection, 64 on two routes on the night highway, and 23 on five routes at the complex intersection. This baseline ground truth consists strictly of per-route cumulative counts without per-vehicle timestamps, which naturally defines the fundamental measurement boundaries of the subsequent count-level metrics.

B. Training Behavior

Fig. 4 shows the training and validation curves. All loss components decrease steadily and settle by the end of training, with the box loss falling from about 1.2 to about 0.85 on the training set and from about 1.1 to 0.85 on validation. Precision and recall rise to plateaus near 0.90 and 0.85, mAP50 reaches about 0.90 and mAP50-95 about 0.70 on the validation split. The gap between the two mean average precision values indicates that localization remains the weaker component under strict overlap thresholds.

The record kept by the experiment tracker makes the plateau precise. Mean average precision over the strict threshold range passes 0.70 at epoch 20 and stands at 0.715 at epoch 45, the last epoch run, so twenty five further epochs buy 0.012. Precision and recall on the validation split settle at 0.883 and 0.838, and the training losses end at 0.886 for the box term, 0.498 for the classification term and 0.999 for the distribution term. One epoch over the training split took about 33 minutes on the graphics processor used, which puts the plateau roughly eleven hours into the run. Two practical consequences follow. An epoch budget of 70 is generous for this dataset, which is why the run was terminated at epoch 45 (Section IV-B), and a study that repeats the adaptation on other camera sets can stop at 30 epochs without losing measurable accuracy.

Fig. 5 relates the F1 score to the confidence threshold per class. The aggregate curve peaks at about 0.86 at a confidence of 0.505, which is the operating point used in the video experiments. Behavior is uneven across classes: car and bus stay high, van and motorcycle are moderate, and truck lies in between. Fig. 6 shows the normalized confusion matrix. The main classes are recognized at rates above 0.80, whereas about 30 per cent of motorcycles are confused with background, which is consistent with their small pixel footprint at night.

C. Image-Level Results

Both test sets carry a four-class vocabulary in which the bus label also covers vans, whereas the adapted model separates the two and the COCO baseline has no van class at all. Tables III and IV therefore list all five classes of the adapted model but can report a score only for the four that have ground truth. The van column distinguishes two different situations: for the baseline it is marked with a dash, because the COCO model has no van class at all, and for the adapted model it is marked as not applicable, because the test sets contain no van

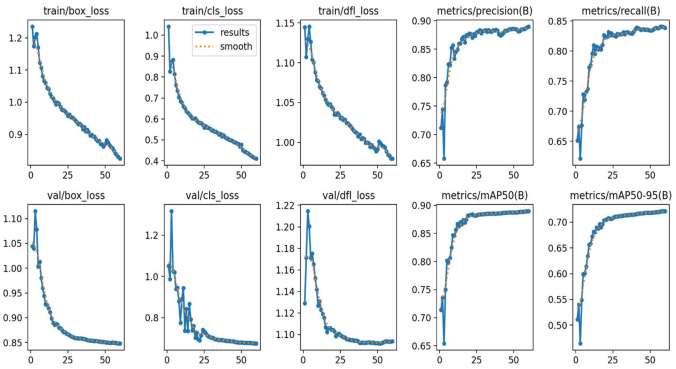


Fig. 4. Training and validation curves of the adapted detector. The upper row shows the loss components and the metrics on the training set, the lower row the same quantities on the validation set

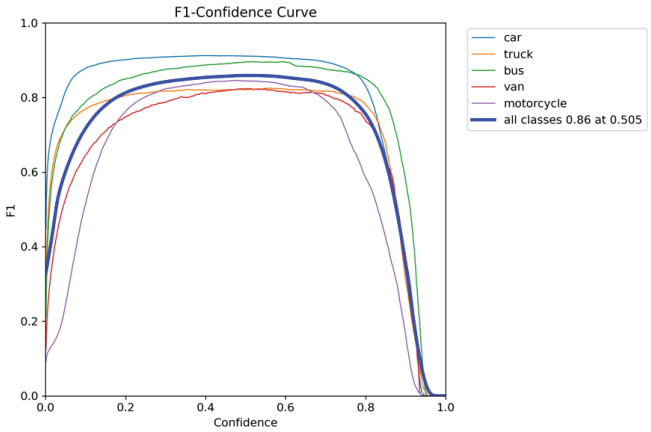


Fig. 5. F1 score against confidence threshold for each vehicle class and for all classes jointly. The aggregate curve peaks at 0.86 at a confidence of 0.505

annotations, so every van prediction of the adapted model is scored against the bus ground truth, where it counts as a miss for the bus class.

Table III reports the combined test set. Fine-tuning improves the F1 score for car from 0.7901 to 0.8048 and for motorcycle from 0.2679 to 0.4372, while the bus score changes from 0.3374 to 0.3257. The truck entry of the adapted model is not reported: the value printed in the review version of this paper was a transcription error, and the correct figure could not be recovered from the evaluation record in time for this version. The aggregate metrics rise as well, mAP50-95 from 0.2604 to 0.2820, mAP50 from 0.4391 to 0.4778 and mAP75 from 0.2814 to 0.2888.

Table IV reports the night test set, where illumination and contrast are lowest. The improvement is larger. Motorcycle detection rises from a complete failure of the baseline to an F1 score of 0.4445, car from 0.7911 to 0.8661, truck from 0.1772 to 0.2446, and mAP50 from 0.3772 to 0.4771.

The one score that drops, bus on the night set from 0.4713 to 0.2421, is a measurement artifact of the vocabulary mismatch rather than a loss of capability, and it affects any comparison between detectors with different vocabularies. Vehicles that

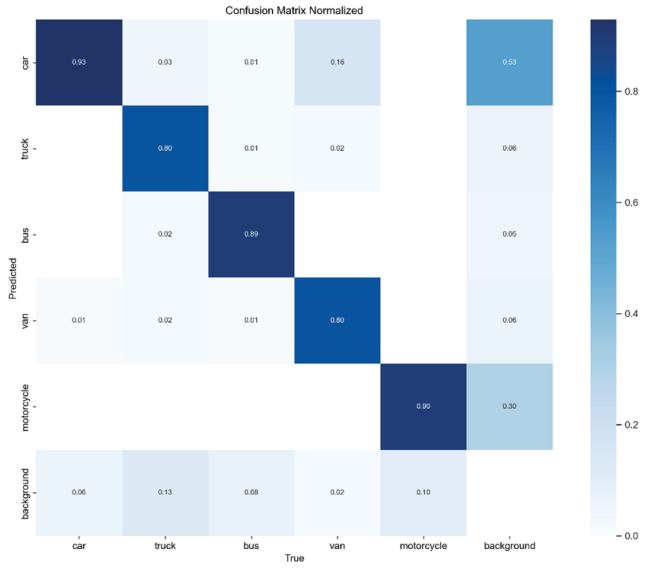


Fig. 6. Normalized confusion matrix of the adapted detector over the five vehicle classes and background

TABLE III. DETECTION PERFORMANCE ON THE COMBINED DATASET

Model	F1 score per class					Mean average precision		
	Car	Truck	Bus†	Van	Motorc.	50-95	50	75
Baseline	0.7901	0.3455	0.3374	–	0.2679	0.2604	0.4391	0.2814
Adapted	**0.8048**	n/r	0.3257	n/a	**0.4372**	**0.2820**	**0.4778**	**0.2888**

Note. –: class not predicted by the model; n/a: no van annotations in the test set; n/r: not reported; †: affected by the bus/van vocabulary mismatch.

the baseline can only call bus are called van by the adapted model and are then scored as misses for the bus class; the baseline, which has no van class, is never penalized in this way. The two bus entries of Table IV, and to a smaller extent of Table III, are therefore not comparable and are marked accordingly. The remedy is to evaluate under a label-space projection φ, applied to predictions and ground truth alike before matching, with $\varphi(\text{bus}) = \varphi(\text{van}) =$ heavy vehicle and the identity elsewhere. Two things follow from the definition of φ without re-running any evaluation. Because the baseline predicts no vans, its projected heavy-vehicle score equals its raw bus score exactly, so φ does not flatter the baseline. For the adapted model, every van prediction that overlaps a van annotated as bus turns from a miss into a match under φ, so the projected score removes precisely the penalty described above and leaves every other error in place. The projected scores themselves require re-scoring both models with merged labels, which was not completed for this version; the protocol is stated here so that the released test sets can be scored under it, and φ is provided as an option in the released evaluation scripts. Until then, the night-set bus row should be read as undefined rather than as a regression, and the aggregate metrics, which include the bus class, as conservative for the adapted model.

TABLE IV. DETECTION PERFORMANCE ON THE NIGHT TEST DATASET

Model	F1 score per class					Mean average precision		
	Car	Truck	Bus†	Van	Motorc.	50-95	50	75
Baseline	0.7911	0.1772	0.4713	–	0.0000	0.2535	0.3772	0.2946
Adapted	**0.8661**	**0.2446**	0.2421	n/a	**0.4445**	**0.2839**	**0.4771**	**0.2997**

Note. –: class not predicted by the model; n/a: no van annotations in the test set; †: metrics are not directly comparable between the two models due to the bus/van vocabulary mismatch.

D. Count-Level Results

Image-level metrics do not tell an operator how many vehicles the system will miss. We therefore report counts per route for the four videos of Section V-A, shown in Fig. 7. For a video with routes $r \in \mathcal{R}$, ground-truth counts G_r and system counts $\hat{N}_r$ obtained from (13), the count recall is

$$\text{CR} = \frac{\sum_{r\in\mathcal{R}} \hat{N}_r}{\sum_{r\in\mathcal{R}} G_r}. \tag{14}$$

Because the ground truth is a per-route count rather than a set of time-stamped events, (14) is a ratio of totals: it measures net undercounting and cannot separate a missed vehicle from a spurious count that happens to offset it. The term count recall is used with this caveat.

Table V gives the per-route counts and the value of (14) per scenario. The adapted detector counts more vehicles than the baseline on every route without exception, and on no route does either model exceed the ground truth. Aggregated over all four videos the count recall rises from 26.1 to 70.1 per cent, a factor of 2.7, which is far larger than the corresponding gain in mAP50 on the night images, a factor of 1.27. This happens because a vehicle only has to be detected often enough for the tracker to hold its identity across the line, so improvements in per-frame recall compound along a trajectory.

The sample is small, 134 ground-truth vehicles, and the uncertainty should be stated. Treating each ground-truth vehicle as a Bernoulli trial, the 95 per cent Wilson score interval is 61.9 to 77.2 per cent for the adapted model (94 of 134) and 19.4 to 34.1 per cent for the baseline (35 of 134). The two intervals are far apart, so the direction and rough size of the improvement are not in doubt despite the small sample, but the point estimate of 70.1 per cent should not be read more precisely than its interval. The interval is also optimistic in one respect: vehicles within a video share lighting, viewpoint and traffic density, so the 134 trials are not independent, and the effective sample is closer to four scenes than to 134 vehicles. The per-scenario values in Table V, which range from 58.1 to 87.0 per cent for the adapted model, show the size of that between-scene variation directly.

The advantage of fine-tuning grows as the scene gets harder. It is smallest at the complex intersection, where the scene is well lit and the two models nearly agree, and largest in fog and on the night highway, where the baseline collapses to a quarter of the true count. The residual errors concentrate in the classes that the confusion matrix identifies as weak, and in the

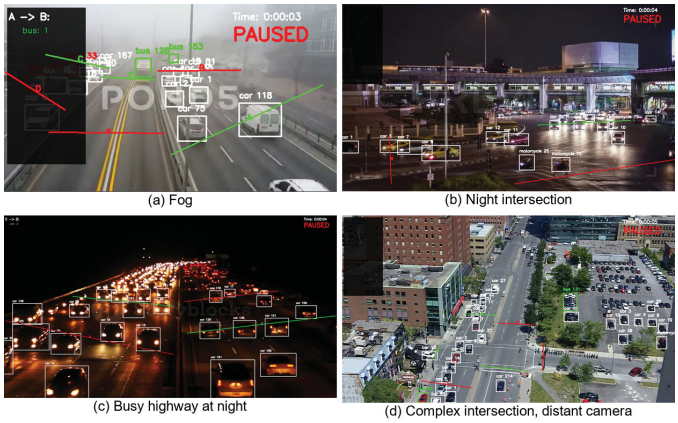


Fig. 7. The four video scenarios used for the count-level evaluation

TABLE V. ROUTE-SPECIFIC VEHICLE COUNTS AND OVERALL RECALL ACROSS VIDEO SEQUENCES

Scenario	Route	Truth	Baseline	Adapted
Fog	A → B	13	0	6
	C → D	15	3	9
	E → X	3	1	3
	count recall	31	*12.9%*	*58.1%*
Night intersection	A → B	10	3	7
	A → C	6	2	4
	count recall	16	*31.3%*	*68.8%*
Highway at night	A → B	30	12	24
	C → D	34	3	21
	count recall	64	*23.4%*	*70.3%*
Complex intersection	C → E	1	1	1
	C → D	6	6	6
	B → D	2	1	2
	B → F	5	2	3
	A → F	9	1	8
	count recall	23	*47.8%*	*87.0%*
All videos		**134**	**26.1%**	**70.1%**

Note. Aggregate values include 95% Wilson score confidence intervals: 19.4–34.1% for the baseline model and 61.9–77.2% for the adapted pipeline.

scenario where the camera stands furthest from the roadway, which shrinks every vehicle and so degrades detection recall and identity stability together.

A word on what the counts do and do not guarantee. The latch of (12) prevents double counting of the same track caused by centroid jitter at a line. It does not strictly prevent spurious counts: a false-positive detection that is tracked for long enough, or a vehicle whose identity is re-assigned after a switch, can traverse both an entry and an exit line and be counted as a route. The counts are therefore not a strict mathematical lower bound on traffic volume, and a spurious route offset by a missed vehicle on the same route would leave Table V unchanged. Empirically, however, no route count exceeds its ground truth on any of the twelve routes, for either model. This is the signature that the analysis of Section IV-D predicts, namely that in these low-visibility scenes the undercounting caused by missed detections and lost identities dominates heavily over false-positive routes, and it supports reading the counts as a practical, though not guaranteed, lower bound. Turning that reading into a measured property requires event-level matching against time-stamped ground truth, which Section VII lists as the first direction of future work.

E. Scope of the Comparison

The experiments evaluate a single detector architecture, YOLO11-m, under two paradigms (COCO-pretrained and fine-tuned) while maintaining an identical tracking algorithm, counting formulation, and operational threshold. This controlled ablation isolates the impact of domain adaptation on both image-level and count-level accuracy by holding all downstream pipeline components constant across the configurations in Tables III to V. Consequently, the observation that count-level improvement doubles image-level gains directly reflects this adaptation effect.

We deliberately restrict the comparison to a single architecture (YOLO11-m) in two states (COCO-pretrained vs. domain-adapted). A broader benchmark including YOLOv8, YOLOv10, or RT-DETR, as well as tracker ablations (e.g., ByteTrack vs. BoT-SORT), would require re-training multiple models and re-running the counting pipeline on identical tracks. Because the counting rule (Algorithm 1), its reference implementation, and the dataset [2], [3] are now formally specified and public, such an architectural benchmark can be conducted by the community without undocumented heuristics. The present work establishes the reproducible baseline for that benchmark, isolating the domain adaptation gain (+44 percentage points in count recall) from architectural differences.

VI. LIMITATIONS, REPRODUCIBILITY AND BROADER IMPACT

A. Limitations

Five limitations bound the claims made here. First, the study covers a single detector family at one scale and one tracker using a single training seed, meaning the reported gains measure domain adaptation rather than architectural or tracking performance. Recent end-to-end architectures that eliminate non-maximum suppression, alongside transformer-based detectors, are not evaluated. Second, one training source and a portion of the evaluation data derive from the same public benchmark, with the partitioning executed by camera location rather than by dataset. A camera-level split excludes shared viewpoints but permits near-duplicate frames of similar scenes, and a perceptual-hash audit across splits (the appropriate validation method) remains unperformed. Since the file lists of all splits are included in the published dataset, this audit can be independently executed, which we consider a prerequisite for utilizing the dataset as a benchmark. Third, the count-level ground truth was established by manual inspection of the four videos without per-vehicle timestamps, so count recall measures net undercounting exclusively. The previously calculated confidence intervals assume vehicle independence, which is violated in practice, preventing error attribution

to individual vehicles or separation into false negatives and spurious counts. Fourth, the counting model operates strictly in image coordinates: without a homography, it yields route and class counters but cannot determine speed or lane occupancy, and it requires a static camera. Fifth, only the detector was profiled, achieving 36.5 ms per frame at a 640-pixel resolution on the training hardware, while the tracking and counting stages were not instrumented. Consequently, the throughput of the complete pipeline and its performance on embedded hardware remain open questions.

B. Reproducibility

Computer vision pipelines for intelligent transportation systems are typically reported as a sequence of design choices; however, critical parameters regarding reproducibility (such as exact loss weights, label mappings, assignment rules, and crossing tests) frequently remain implicit. This paper defines each component in closed form: the objective function spans (3) to (6), the class mapping and target projections are integrated into the data definition, and the crossing predicate is formalized in (9). Two practices ensured experimental recovery across diverse hardware environments: deploying an experiment tracker to preserve the hyperparameter namespace, per-epoch scalars, and model checkpoints independently of the host machine, and publishing the fully assembled dataset alongside the evaluation code. We advocate both approaches as a baseline standard for the field, given that a described pipeline lacking accessible checkpoints and scripts constitutes an unverified claim rather than a reproducible result.

The domain adaptation pipeline was implemented within a standard deep learning environment using a single NVIDIA GeForce RTX 3060 Ti GPU (8 GB), requiring approximately 33 minutes per training epoch. The exact software dependencies and version specifications are provided in the public repository [3]. The assembled dataset of 43,403 annotated images is openly accessible [2].

To ensure the reproducibility of the counting stage and the evaluation protocol, we publicly release the exact implementation of Algorithm 1, the interactive line-marking utility, the class-mapping configuration, and the label-space projection φ in our open-source repository [3]. The two image test sets with their annotations, alongside the four video segments, scene descriptions, and per-route ground truth, are currently being integrated into the same repository in compliance with the redistribution terms of their original sources. Prior to the completion of this upload process, these verification assets are available directly from the authors.

C. Deployment Considerations and Privacy

Camera-based traffic monitoring inevitably captures potentially identifiable information, including vehicle configurations, license plates, and incidental pedestrians or cyclists. To mitigate privacy risks, this study processes and reports exclusively aggregate, class-level statistics (such as counts, routing distributions, and detection metrics) rather than individual trajectories. Operational deployments should establish strict data-retention policies, restrict video storage to the minimum required for flow analysis, and anonymize faces and license plates in compliance with local regulations. Furthermore, because the foundational datasets originate from specific regions and camera geometries, the model's generalization may vary in environments with distinct vehicle distributions, road markings, or lighting. System deployers must therefore validate performance on local footage before integrating counts into traffic-control decisions. This validation is critical given the lower baseline accuracy observed for trucks and motorcycles, as systematic undercounting of minor classes remains challenging to detect downstream.

VII. Conclusion

This paper presented a robust traffic monitoring pipeline for low-visibility conditions. Alongside the adapted detector, we introduced the underlying fine-tuning dataset [2], a formal training objective, a mathematically precise counting stage, and open-sourced the counting and evaluation code [3]. Fine-tuning a medium-scale YOLO11 detector on 43,403 harmonized images raised the night test set mAP50 from 0.377 to 0.477. This adaptation restored motorcycle detection capabilities, which the baseline COCO model failed to capture. The single apparent regression, involving the bus category on the night dataset, stemmed from a vocabulary mismatch between the detector and test annotations. We resolved this by specifying a label-space projection φ; the released evaluation scripts compute these projected scores directly on the released test sets.

Furthermore, formalizing the counting logic as a two-sided orientation test on centroid trajectories yielded a frame-rate independent decision rule. This formulation incorporates a half-plane convention for degenerate cases, a per-track latch mechanism, and explicit route counters. Operationally, it requires only four cross products per track and line. This transparency ensures that system failures are directly attributable to either the tracker or the detector rather than undocumented heuristics. Evaluated on four real-world video segments, the adapted pipeline increased overall count recall from 26.1% to 70.1%, with a 95% confidence interval spanning 61.9% to 77.2%. Crucially, no evaluated route exceeded the ground truth count. This observation remains consistent with the systematic undercounting bias predicted by our theoretical analysis, supporting the interpretation of these counts as a practical lower bound.

Three future research directions follow from these results. The first is a count-level benchmark featuring time-stamped, per-vehicle ground truth. This will replace manual video counting with a reproducible validation procedure, enabling researchers to measure counting precision alongside recall and experimentally confirm the predicted lower-bound property. The second direction involves comparative evaluations that the present ablation deliberately excluded. These include testing alternative detector families on the identical dataset, deploying ByteTrack and BoT-SORT on the same sequences, and benchmarking the orientation test against proximity bands on

identical trajectories. Because our counting rule is defined in closed form and the source code is publicly available [3], such comparisons can isolate the impact of individual pipeline components without relying on undocumented heuristics. Finally, the taxonomy-aligned evaluation protocol should be consistently applied to report projected scores whenever models with divergent vocabularies are compared. Without this adjustment, fine-grained classes refined by one architecture but absent in the evaluation set are silently penalized.

REFERENCES

[1] G. Jocher and J. Qiu, *Ultralytics YOLO11*, version 11.0.0, 2024. Web: https://github.com/ultralytics/ultralytics.

[2] V. Anisimov, *Traffic Dataset 43k: Low-Visibility Vehicle Detection*, Kaggle, 2025. Web: https://www.kaggle.com/datasets/vladislavanisimov123/traffic-dataset440k.

[3] V. Anisimov, O. Pereziabov, and I. Afanasyev, "Night Traffic Monitoring: Counting Module and Evaluation Scripts," GitHub repository, 2026. [Online]. Web: https://github.com/afanasyspb/night-traffic-monitoring

[4] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "YOLOv10: Real-time end-to-end object detection," in *Advances in Neural Information Processing Systems*, vol. 37, 2024.

[5] M. A. R. Alif, "YOLOv11 for vehicle detection: Advancements, performance, and applications in intelligent transportation systems," *arXiv preprint arXiv:2410.22898*, 2024.

[6] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: Challenges, architectural successors, datasets and applications," *Multimedia Tools and Applications*, vol. 82, no. 6, pp. 9243–9275, 2023.

[7] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, "YOLOv8 to YOLO11: A comprehensive architecture in-depth comparative review," *arXiv preprint arXiv:2501.13400*, 2025.

[8] J. Zhu, X. Li, P. Jin, Q. Xu, Z. Sun, and X. Song, "MME-YOLO: Multi-sensor multi-level enhanced YOLO for robust vehicle detection in traffic surveillance," *Sensors*, vol. 21, no. 1, p. 27, 2020.

[9] Y. Miao, F. Liu, T. Hou, L. Liu, and Y. Liu, "A nighttime vehicle detection method based on YOLO v3," in *2020 Chinese Automation Congress (CAC)*, 2020, pp. 6617–6621.

[10] S. Sooksatra, T. Kondo, P. Bunnun, and A. Yoshitaka, "Headlight recognition for night-time traffic surveillance using spatial-temporal information," *Signal, Image and Video Processing*, vol. 14, no. 1, pp. 107–114, 2020.

[11] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "ByteTrack: Multi-object tracking by associating every detection box," in *European Conference on Computer Vision*, 2022, pp. 1–21.

[12] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "BoT-SORT: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022.

[13] T. Mudavath and A. Mamidi, "Object detection challenges: Navigating through varied weather conditions, a comprehensive survey," *Journal of Ambient Intelligence and Humanized Computing*, vol. 16, no. 2, pp. 443–457, 2025.

[14] M. Sualeh and G.-W. Kim, "Visual-LiDAR based 3D object detection and tracking for embedded systems," *IEEE Access*, vol. 8, pp. 156285–156298, 2020.

[15] M. Trigka and E. Dritsas, "A comprehensive survey of machine learning techniques and models for object detection," *Sensors*, vol. 25, no. 1, p. 214, 2025.

[16] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, vol. 193, p. 102907, 2020.

[17] *JRDR Deraining Dataset*, Kaggle, 2025. Web: https://www.kaggle.com/datasets/shivakanthsujit/jrdr-deraining-dataset.

[18] V. S. Patel, K. Agrawal, and T. V. Nguyen, "A comprehensive analysis of object detectors in adverse weather conditions," in *2024 58th Annual Conference on Information Sciences and Systems (CISS)*, 2024, pp. 1–6.

[19] X. Huang, P. Wang, X. Cheng, D. Zhou, Q. Geng, and R. Yang, "The ApolloScape open dataset for autonomous driving and its application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2702–2719, 2020.

[20] G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker, and C. V. Jawahar, "IDD: A dataset for exploring problems of autonomous navigation in unconstrained environments," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 1743–1751.

[21] C. Sakaridis, D. Dai, and L. Van Gool, "ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding," in *Proc. IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10745–10755.

[22] B. T. Tran-Le, V. Patel, V. T. Huynh, M. K. Tran, K. Agrawal, M. T. Tran, and T. V. Nguyen, "Towards safer roads: Benchmarking object detection models in complex weather scenarios," *Machine Vision and Applications*, vol. 36, p. 94, 2025.

[23] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digital Signal Processing*, vol. 126, p. 103514, 2022.

[24] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision (ECCV)*, Springer, 2014, pp. 740–755.

[25] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[26] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving video database with scalable annotation tooling," *arXiv preprint arXiv:1805.04687*, 2018.

[27] *Road Vehicle Images Dataset*, Kaggle, 2025. Web: https://www.kaggle.com/datasets/ashfakyeafi/road-vehicle-images-dataset.

[28] New Workspace, *Training-1-2-3-4*, Open Source Dataset, Roboflow Universe, 2022. Web: https://universe.roboflow.com/new-workspace-7ucki/training-1-2-3-4.

[29] Archie Junio, *Car Detection Model*, Computer Vision Model, Roboflow Universe, 2025. Web: https://universe.roboflow.com/archie-junio-dxv5t/car-detection-model-bwjpb/dataset/17.

[30] *Vehicle Detection*, Computer Vision Dataset, Roboflow Universe, 2023. Web: https://universe.roboflow.com/lynkeus03/vehicle-detection-by9xs/dataset/3.

[31] *Traffic Intersection Vehicle Detection*, Computer Vision Model, Roboflow Universe, 2023. Web: https://universe.roboflow.com/vai/traffic-intersection-vehicle-detection/dataset/1.

[32] *Intersection Dataset*, Open Source Dataset, Roboflow Universe, 2023. Web: https://universe.roboflow.com/thesis-8yadw/intersection-t5pp2.

[33] *Vehicle Day Night Dataset*, Roboflow Universe, 2024. Web: https://universe.roboflow.com/vehicleseperate/vehicle-day-night/dataset/3.

[34] European Automobile Manufacturers Association, *Vehicles in Use, Europe 2023*, 2023. Web: https://www.acea.auto/files/ACEA-report-vehicles-in-use-europe-2023.pdf.

[35] *Ultralytics YOLO11 Documentation*, Ultralytics, 2024. Web: https://docs.ultralytics.com/models/yolo11/.

[36] W. Lv, Y. Zhao, Q. Chang, K. Huang, G. Wang, and Y. Liu, "RT-DETRv2: Improved baseline with bag-of-freebies for real-time detection transformer," *arXiv preprint arXiv:2407.17140*, 2024.

[37] *RT-DETRv2 vs YOLO11: Technical Comparison*, Ultralytics, 2024. Web: https://docs.ultralytics.com/compare/rtdetr-vs-yolo11/.

[38] *ClearML, Your Entire MLOps Stack in One Open-Source Tool*, ClearML, 2024. Web: https://clear.ml/.

[39] C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang, "TOOD: Task-aligned one-stage object detection," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 3490–3499.

[40] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 34, no. 7, 2020, pp. 12993–13000.

[41] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Yang, and J. Tang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 21002–21012.

[42] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.