Real-Time Traffic Congestion Detection with Fine-Tuned YOLOv8s

Erkan Işık Bacak
Bahçeşehir University
erkan.bacak@bahcesehir.edu.tr

May 10, 2025

Abstract

We present a real-time traffic-scene analytics pipeline that detects and quantifies vehicle congestion on road segments using a custom-fine-tuned YOLOv8s detector. The model is trained on a 6 903-image dataset (5 805 train, 549 val, 249 test) from Roboflow and achieves $\mathbf{mAP}_{50} = \mathbf{0.956}$ and $\mathbf{mAP}_{50-95} = \mathbf{0.825}$. A lightweight temporal smoother then converts per-frame detections into a continuous congestion score, while pedestrian counts are handled separately for potential cross-walk logic. Experiments on 720p CCTV video demonstrate that our Python runtime runs comfortably above real-time (∼138 fps) on double NVIDIA T4 GPU, processing each frame in 7.2 ms approximately.

1 Related Work

Vision-based vehicle congestion detection has progressed through several stages, from classical image processing to modern deep learning frameworks.

Classical and CNN-Based Methods Early systems relied on background subtraction and optical flow to count vehicles and infer congestion by thresholding vehicle counts or speed estimates [**ren2015fasterrcnn**, 5]. With deep learning's rise, *image classification* models (e.g., AlexNet, GoogLeNet) were trained to recognize congested vs. free-flow states directly from frames, demonstrating the feasibility of data-driven congestion classification [5]. More specialized CNN architectures followed: for instance, Cui *et al.* introduced a multi-branch network that first detects vehicles, then aggregates their features for a dedicated congestion classifier, improving robustness under varied viewpoints [4]. Luo *et al.* proposed a Small Parallel Residual CNN (SPRCNN) that achieves high accuracy with significantly fewer parameters, highlighting the importance of lightweight models for real-time traffic applications [2].

Transformer-Based Approaches Transformers brought *global context* via self-attention to object detection. DETR (Detection Transformer) by Carion *et al.* removed hand-crafted post-processing and directly predicted bounding boxes and classes in an end-to-end manner, matching Faster R-CNN's accuracy while capturing long-range dependencies in traffic scenes [1]. Deformable DETR [6] improved convergence speed and small-object detection by attending only to a sparse set of learned offsets, making transformers more practical for vehicle detection in cluttered environments.

One-Stage Detectors and YOLO Variants One-stage detectors like YOLO remain dominant for real-time traffic analysis. YOLOv1 introduced the single-shot paradigm, achieving 45 fps on GPUs by unifying localization and classification into one network pass [3]. Subsequent YOLO versions (v2–v8) incorporated multi-scale heads, advanced backbones, and training augmentations (e.g., mosaic), yielding a family of models that trade off throughput and accuracy [**bao2023yolo5traffic**]. These detectors are

routinely used to count vehicles per frame and derive congestion indices from object densities.

Discussion and Our Contribution While CNNs and transformers excel in accuracy, they often impose high computational costs or require separate classification stages. YOLO-based methods deliver the throughput needed for on-the-fly analysis but traditionally treat all vehicles equally. Our work fine-tunes YOLOv8s—leveraging its compact architecture and fast convergence—to perform vehicle detection, then computes a class-weighted congestion score (e.g., buses weighted higher than cars) and applies a sliding-window smoother. This design unites the real-time capabilities of one-stage detectors with domain-specific scoring to yield a responsive, resource-efficient congestion monitoring system.

2 Method

Figure 1 shows the end-to-end pipeline. We first fine-tune YOLOv8s on a labelled image set (§2.1). Inference frames flow through the detector, producing bounding boxes that are class-filtered and aggregated into a scalar congestion score (§2.2). A short FIFO history smooths the score before rendering.

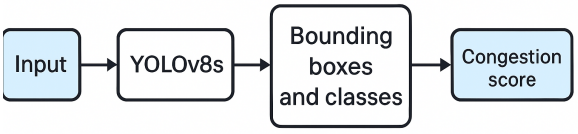

Figure 1: System pipeline. A FIFO buffer of the last k frames stabilises the congestion metric.

2.1 Model Training

Listing 1 shows the training script. Training consumed 210 minutes (approx. 2 minutes per epoch) on a Google Colab T4 GPU with ˜16 GB of memory. After 90 epochs the dataloader mosaic was closed which boosted the model's performance on training set. Checkpoints and TensorBoard logs are stored under `yolov8s_custom_finetune2`.

Listing 1: Training script

```
model.train(
    data='/CCTV_dataset/data.yaml',
    epochs=100,
    imgsz=640,
    batch=32,
    name='yolov8s_custom_finetune2',
    mosaic=0.5,
    mixup=0.1,
    hsv_h=0.015, hsv_s=0.7, hsv_v=0.4,
    translate=0.1, scale=0.3,
    fliplr=0.5, flipud=0.0,
    degrees=0.0, patience=50,
    amp=True)
```

2.2 Congestion Scoring

Each detected instance receives a weight w_c based on class c (*bicycle*=0.5, *bus*=2.0, *car*=1.0, *motorbike*=1.0). The per-frame score is

$$S_t = \sum_c n_{c,t}\, w_c.$$

We keep two sliding windows of length $k = 5$ frames: one for vehicles $\{S_{t-4}, \ldots, S_t\}$ and one for pedestrian counts $\{P_{t-4}, \ldots, P_t\}$. The displayed metrics are their arithmetic means:

$$\tilde{S}_t = \frac{1}{5}\sum_{i=0}^{4} S_{t-i}, \quad \tilde{P}_t = \frac{1}{5}\sum_{i=0}^{4} P_{t-i}.$$

3 Data Description

We used the Street View GDOGO dataset from Roboflow[1], with splits of 5 805 training images, 549 validation images, and 249 test images.

Table 1: Dataset splits.

Split	Images
Train	5 805
Val	549
Test	249

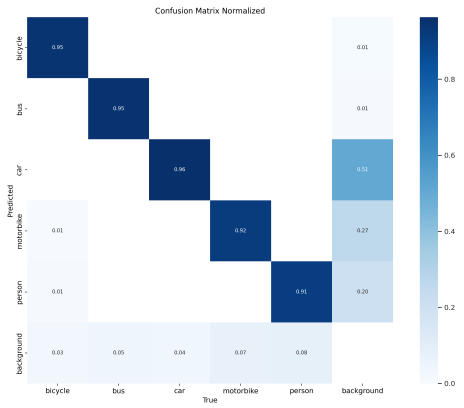

Figure 3: Normalized Confusion Matrix.

4 Experiments and Results

Training graphs are in Figure 2. The final checkpoint reaches $\mathbf{mAP}_{50} = \mathbf{0.956}$ and $\mathbf{mAP}_{50\text{-}95} = \mathbf{0.825}$. On 720p test footage, end-to-end processing (including smoothing and OpenCV drawing) runs at ∼138 fps (0.3 ms preprocess, 3.6 ms inference, 3.3 ms postprocess per image).

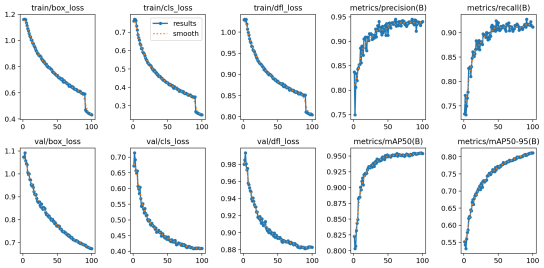

Figure 2: Loss and mAP during training.

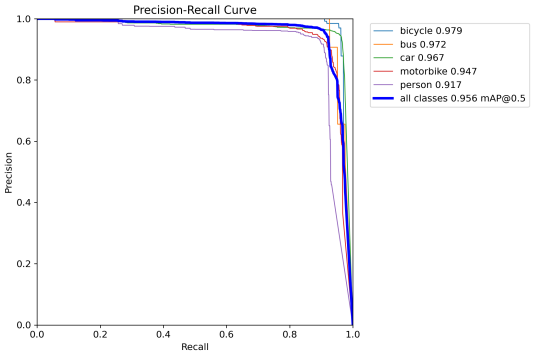

Figure 4: Precision-Recall curve.

Table 2: Key hyper-parameters.

Parameter	Value
Learning rate	0.001111 (auto-determined AdamW)
Optimiser	AdamW, momentum 0.9
Weight decay	0.0005
Batch size	32
Smoothing window k	5 frames
Vehicle weights w	bicycle 0.5, bus 2.0, car 1.0, motorbike 1.0

[1] `https://universe.roboflow.com/fsmvu/street-view-gdogo/dataset/1`

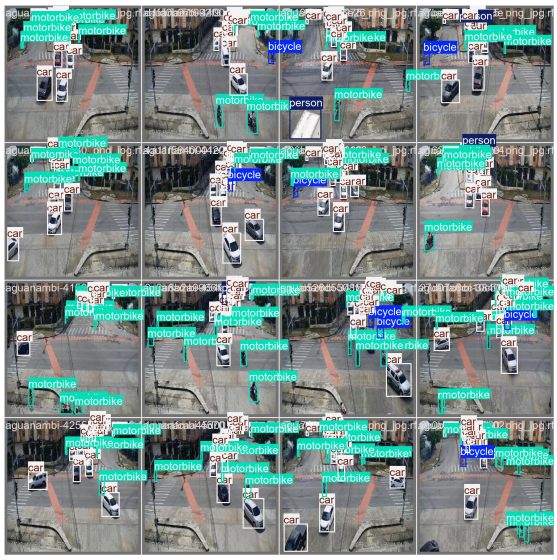

Figure 5: Model Results on unseen data.

5 Conclusion

In this work we deliver a compact yet accurate system for real time congestion monitoring. We used a finte-tuned the YOLOv8s with a dataset which consists of cctv images from Turkey suitable for object detection, including 5 labels: Person, car, bicycle, motorbike, bus.

Supplementary Material

- Full training and inference scripts available at: `https://github.com/ErkanIsikB/traffic`$_{c}ongestion_{a}i$

References

[1] Nicolas Carion et al. "End-to-end object detection with transformers". In: *European Conference on Computer Vision (ECCV)*. 2020, pp. 213–229.

[2] Yuan Luo et al. "Small Parallel Residual Convolutional Neural Network for Image-Based Traffic Congestion Detection". In: *Scientific Reports* 14.1 (2024), p. 1234.

[3] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[4] Zhaohui Yang et al. "Deep learning-based congestion detection at urban intersections". In: *Sensors* 21.6 (2021), p. 2122.

[5] Yu Zhang et al. "FDS-CNN: An end-to-end deep learning framework for urban traffic flow prediction using remote sensing imagery". In: *Remote Sensing* 9.6 (2017), p. 556.

[6] Xizhou Zhu et al. "Deformable DETR: Deformable transformers for end-to-end object detection". In: *International Conference on Learning Representations (ICLR)*. 2021.