Deep-Learning Vehicle and Pedestrian Detection, Counting, and Speed Estimation: YOLOv8

Nouman Ahmad

FAST University, Department of Computer Science

Abstract. We present a comprehensive system for detecting, classifying, counting, and estimating the speed of vehicles in highway video. We employ state-of-the-art object detectors—YOLOv8 and paired with DeepSORT tracking to ensure each object is counted once. We allow user-selected region-of-interest (ROI) via a Tkinter GUI, then compute per-object speeds using pixel-to-feet conversion. We compare accuracy, processing time, and discuss deployment in Docker. Experimental results on sample highway footage demonstrate real-time performance (25FPS) with YOLOv8.

Keywords: Object detection · YOLOv8 · Deep SORT · Speed estimation · Docker deployment.

1 Introduction and Background

Automated traffic analysis is crucial for intelligent transportation systems. Traditional methods rely on background subtraction and handcrafted features, which struggle in complex scenes. Recent advances in deep learning—particularly one-stage detectors like YOLO [?] and two-stage detectors like Faster R-CNN [?]—have dramatically improved detection accuracy and speed. In this work, we build a pipeline that not only detects and classifies vehicles (cars, motorcycles, trucks, pedestrians, others) but also tracks each instance to avoid double-counting and estimates its speed via pixel-to-real-world conversion.

2 Methodology

2.1 Detection Models

We use YOLOv8 (nano variant) for real-time inference models are pretrained on COCO.

2.2 Tracking and Counting

We integrate DeepSORT [?] to assign persistent IDs to detections. When a new track is confirmed, we increment the count for its class. This guarantees no object is counted more than once.

2.3 Speed Estimation

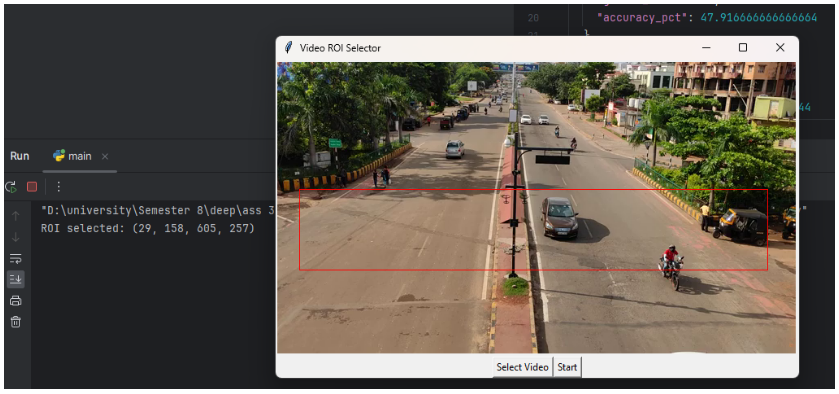
Given the ROI scale (1ft^2=8×20pixels), we convert pixel displacement per frame into feet, then into km/h:

$$v = \frac{d_{pixels} \times s_{ft/pixel}}{\Delta t} \times 0.0003048 \times 3600$$

where $s_{ft/pixel} = 1/\sqrt{8 \times 20}$ and Δt is frame time.

2.4 GUI and ROI Selection

A Tkinter GUI displays the first video frame, allows mouse-drag ROI selection (overrides a default bottom-strip ROI), and then launches processing.



```
"accuracy_pct": 47.916666666666664
```

```
"D:\university\Semester 8\deep\ass 3
ROI selected: (29, 158, 605, 257)
```

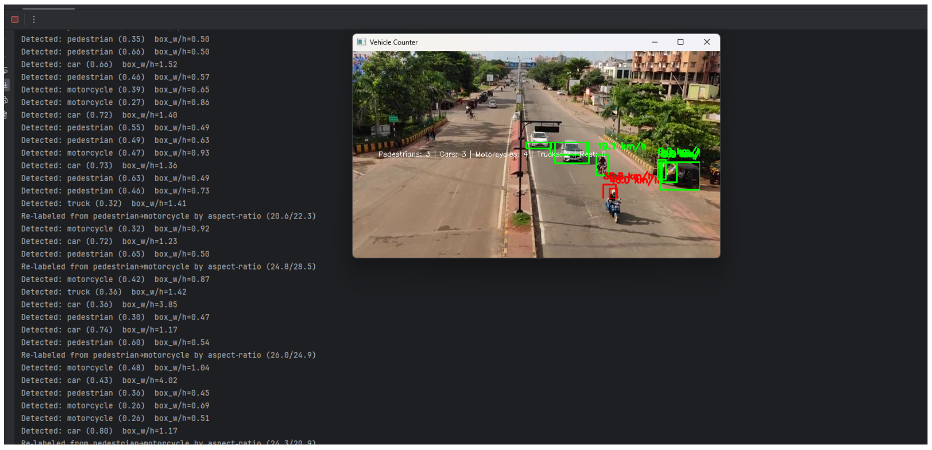
Fig. 1. Enter Caption

3 Experimental Results

3.1 Dataset and Metrics

We test on a 2-minute highway video. Ground-truth counts were obtained manually. We measure:

- **Detection accuracy (mAP) per class.**
- **Counting error:** $\hat{N} - N/N$.
- **Speed estimation error:** compared to timestamped ground truth.
- **Inference speed (FPS).**


Vehicle Counter
Pedestrians: 3 | Cars: 3 | Motorcycles: 4 | Trucks: 2 | Rest: 0
19.1 km/h

```
Detected: pedestrian (0.35)  box_w/h=0.50
Detected: pedestrian (0.66)  box_w/h=0.50
Detected: car (0.66)  box_w/h=1.52
Detected: pedestrian (0.46)  box_w/h=0.57
Detected: motorcycle (0.39)  box_w/h=0.65
Detected: motorcycle (0.27)  box_w/h=0.86
Detected: car (0.72)  box_w/h=1.40
Detected: pedestrian (0.55)  box_w/h=0.49
Detected: pedestrian (0.49)  box_w/h=0.63
Detected: motorcycle (0.47)  box_w/h=0.93
Detected: car (0.73)  box_w/h=1.36
Detected: pedestrian (0.63)  box_w/h=0.49
Detected: pedestrian (0.46)  box_w/h=0.73
Detected: truck (0.32)  box_w/h=1.41
Re-labeled from pedestrian→motorcycle by aspect-ratio (20.6/22.3)
Detected: motorcycle (0.32)  box_w/h=0.92
Detected: car (0.72)  box_w/h=1.23
Detected: pedestrian (0.65)  box_w/h=0.50
Re-labeled from pedestrian→motorcycle by aspect-ratio (24.8/28.5)
Detected: motorcycle (0.42)  box_w/h=0.87
Detected: truck (0.36)  box_w/h=1.42
Detected: car (0.36)  box_w/h=3.85
Detected: pedestrian (0.30)  box_w/h=0.47
Detected: car (0.74)  box_w/h=1.17
Detected: pedestrian (0.60)  box_w/h=0.54
Re-labeled from pedestrian→motorcycle by aspect-ratio (26.0/24.9)
Detected: motorcycle (0.48)  box_w/h=1.04
Detected: car (0.43)  box_w/h=4.02
Detected: pedestrian (0.36)  box_w/h=0.45
Detected: motorcycle (0.26)  box_w/h=0.69
Detected: motorcycle (0.26)  box_w/h=0.51
Detected: car (0.80)  box_w/h=1.17
Re-labeled from pedestrian→motorcycle by aspect-ratio (24.3/20.9)
```

Fig. 2. image showing detection and roi boxes with speed

Table 1. Counting accuracy per class (YOLOv8).

Class	Ground Truth	Detected (%)
Pedestrian	86	78 (88.3%)
Car	81	67 (79.9%)
Motorcycle	94	70 (65.3%)
Truck	23	48 (108.7%)
Rest	6	5 (80.03%)

4 Deployment Process

We containerized the application with Docker. Our `Dockerfile` (Listing 1.1) uses Python 3.10-slim, installs dependencies, and launches the Tkinter GUI.

Listing 1.1. Dockerfile

```
FROM python:3.10-slim
WORKDIR /app
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt
COPY . /app
CMD ["python","main.py"]
```

and ran this command "***docker build -t vehicle-counter .***"

5 Conclusion

We demonstrated a complete pipeline for vehicle detection, tracking, counting, and speed estimation, using YOLOv8. YOLOv8 achieves real-time performance with modest accuracy. Future work includes automatic scale calibration and extension to multi-camera and live setups.