

# Optimized zone-based vehicle speed estimation and classification

*Masoud Vahid Dastgerdi\**, *Andrey Leus*, *Viktor Zuev*, *Nikita Vodichev*, *Ivan Kholodnyak*, *Vladislav Efremov*, *Daniil Mangazeev*, and *Nikita Pecherkin*

Moscow Institute of Physics and Technology (MIPT), Dolgoprudny, Russia

**Abstract.** This article pioneers the fusion of advanced computer vision, and environmental science in order to be a starting point in ecological tasks and environmental benefits. Utilizing state-of-the-art tools like YOLOv7 and innovative algorithms, the study achieves unmatched accuracy in vehicle identification, classification, tracking, and speed analysis. By optimizing YOLOv7-e6e-1280 architecture using TensorRT and reduced precision, real-time analysis becomes possible without compromising accuracy. The integration of the Vanishing Point Principle for road zoning and zone-based speed calculation provides nuanced insights into driving behaviors. Detailed vehicle classification and robust tracking offer valuable data for urban planning and ecological studies. This approach increases our potential in vehicular analysis, setting new standards for research in urban development, transportation, and environmental science.

## 1 Introduction

In this study, we explore the intersection of advanced computer vision, machine learning, and environmental science to tackle the task of calculating vehicle speeds on roads; Leveraging tools such as mmPretrain [1], we achieve vehicle classification within various environments. The integration of the Bot-Sort algorithm enables real-time vehicle tracking, providing data for understanding vehicle behaviors. The Bot-Sort algorithm [2] uses Kalman filtering [3] which has been mentioned in this article but you can investigate the modifications in the methodology.

Our approach incorporates the YOLOv8 object detection model, trained on a diverse dataset of traffic cone images, enhancing road analysis and enabling efficient zone creation. By combining the Vanishing Point Principle from projective geometry with traffic cone detection, we swiftly and accurately reconstruct road perspectives, defining zones for speed analysis. In these articles [4-6] some similar approaches have been conducted but as far as we need absolute accuracy, we had to implement a method to find road sides by traffic cones.

Within these defined zones, mathematical algorithms are applied to calculate vehicle speeds, offering nuanced insights into driving behaviors. One important factor in this project is speed in processing videos and tracking algorithm which was accomplished by

---

\* Corresponding author: [masoud.vahid10@gmail.com](mailto:masoud.vahid10@gmail.com)

modification in Kalman filter in C++. And now by bridging computer vision and mathematics, we open new avenues for understanding vehicular dynamics and their ecological impact.

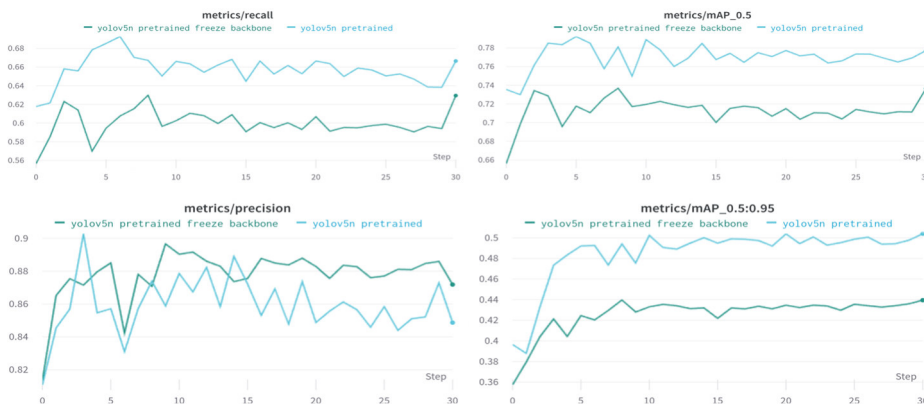
## 2 Methodology

Our methodology integrates computer vision algorithms and geometrical techniques to tackle the challenge of understanding vehicular speed and types.

### 2.1 Computer Vision Techniques and Algorithms

**Car Detection:** In this study, our primary focus was on utilizing two versions of the YOLO (You Only Look Once) object detection model, namely YOLOv5 and YOLOv7 [7], to detect cars. To select the optimal architecture, a series of experiments were conducted comparing YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5x, and YOLOv5l. During the experiments, image resolutions, augmentations, and backbone freezing were varied. All experiments were performed on a reduced dataset of 10,000 images. The goal was to find the best balance between detection quality and video stream processing speed. Increasing the architecture size and input resolution generally improved object detection quality but reduced the neural network's processing speed. Given the plan for simultaneous processing of multiple streams on a single GPU, this speed was critical.

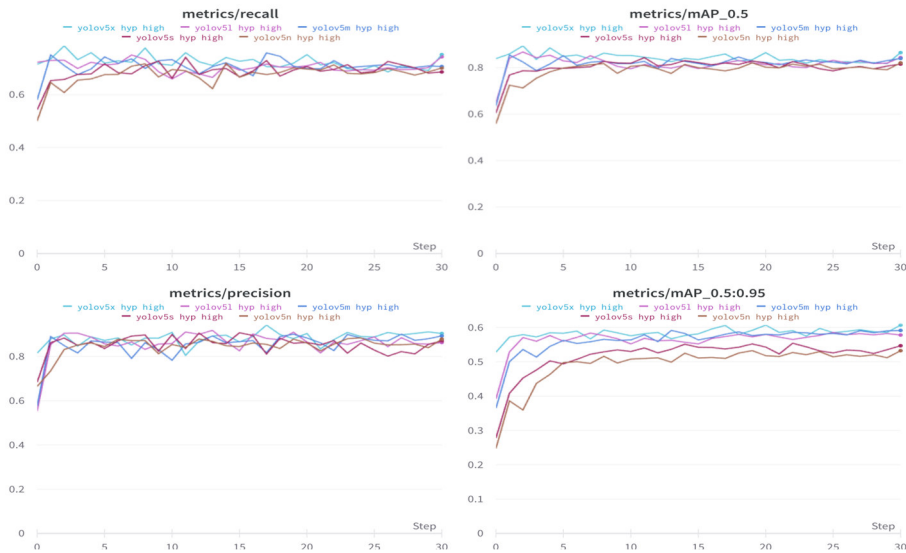
An experiment involving "backbone" freezing was conducted on the YOLOv5n architecture. The resulting graphs from this experiment are shown in Figure 1.



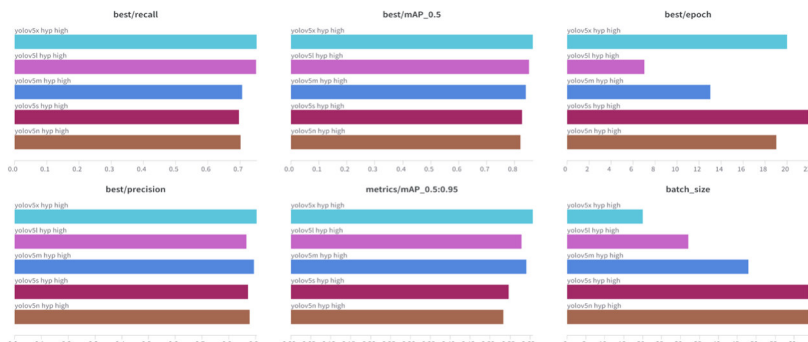
**Fig. 1.** Training Metrics with "Backbone" Freezing.

From the graphs (Figure 1), it is evident that the YOLOv5n architecture does not overfit the data. The best epoch is the last one. Since there is no overfitting, freezing the "backbone" does not improve the training quality.

A comparative experiment was performed with different augmentations such as Mosaic [8] at a resolution of 1920. The experiment results are presented in Figures 2 and 3.



**Fig. 2.** Training Graphs of YOLOv5 Models.



**Fig. 3.** Metrics of YOLOv5 Models.

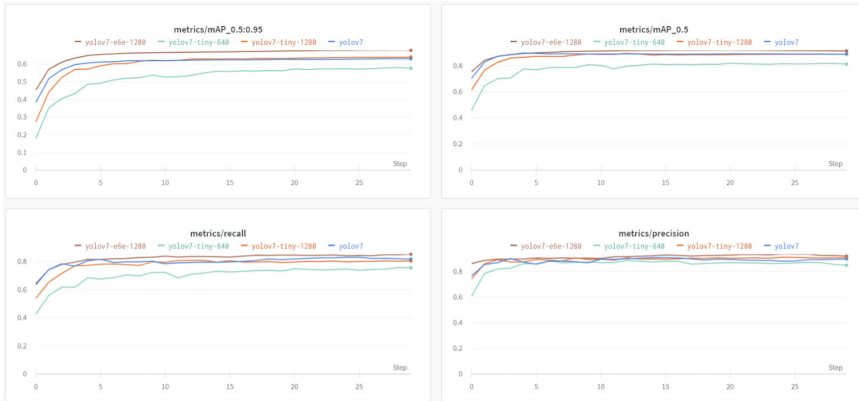
From the graphs (Figure 2), it is observed that as the architecture size increases, the metrics improve. Additionally, it can be concluded that the YOLOv5x architecture does not overfit the dataset containing 10,000 images with a resolution of 1920.

According to the authors of a referenced article [7], YOLOv7 outperforms YOLOv5 in terms of speed and accuracy. Comparative detector models are presented in Table 1.

**Table 1.** Comparative Table of Detectors for Yolov7 and yolov5.

Model	#Param	FLOPs	Size	FPS	AP test	AP val
YOLOv5-n	1.9M	4.5G	640	159	-	28.0%
YOLOv5-s	7.2M	16.5G	640	156	-	37.4%
YOLOv5-m	21.2M	49.0G	640	122	-	45.4%
YOLOv5-l	46.5M	109.1G	640	99	-	49.0%
YOLOv5-x	86.7M	205.7G	640	83	-	50.7%
YOLOv7-tiny	6.2M	13.8G	640	<b>286</b>	38.7%	38.7%
YOLOv7	36.9M	104.7G	640	161	51.4%	51.2%
YOLOv7-X	71.3M	189.9G	640	114	<b>53.1%</b>	<b>52.9%</b>

Training matrices of YOLOv7 models are presented in Figure 4. During this experiment, architectures YOLOv7 and YOLOv7-tiny were compared at resolutions 640 and 1280.

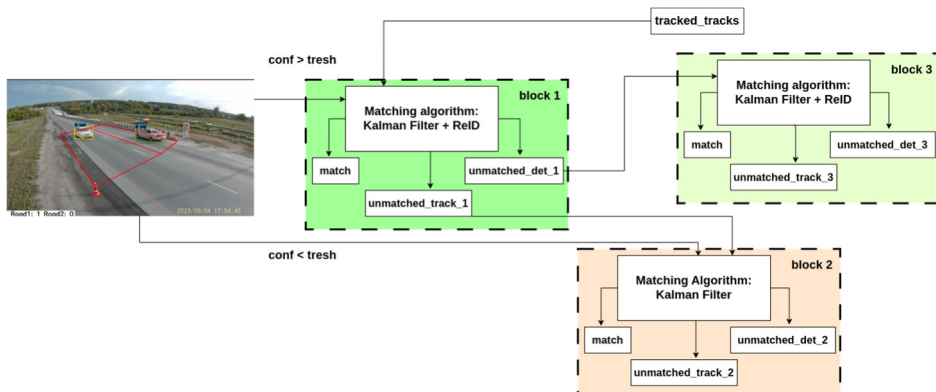


**Fig. 4.** Training Graphs of YOLOv7 Models.

**Car Classification:** In our research, we utilize mmPretrain to refine the classification process, employing the algorithm’s capability to differentiate between diverse vehicle types effectively. Our classification system categorizes vehicles into 7 distinct types: cars, low-weight trucks, trucks exceeding 12 tons, vans, buses, electric transports and motorbike. These detailed classifications not only enrich our immediate analysis but also serve as valuable data for subsequent ecological studies. By providing information about vehicles, our methodology lays a foundation for future ecological research, facilitating a deeper understanding of environmental impact and urban planning.

**Tracking Algorithm:** The tracking algorithm (Figure 5) processes two types of detections - low-confidence and high-confidence, the former having a detector confidence below a certain threshold and the latter comprising the rest. High-confidence detections undergo re-identification [9], reducing the load on the re-identifier. Both types of detections are then fed into respective blocks of our matching algorithm.

The algorithm computes pairwise Intersection over Union [10] (IoU) intersections for bounding boxes and cosine distances between feature vectors of detections and averaged feature vectors of tracks from previous detections. For high-confidence detections, Kalman Filter predictions are utilized to enhance accuracy. The algorithm then utilizes the Hungarian algorithm [11] to match detections and tracks, considering IoU values above a specified threshold. Unmatched tracks and detections are segregated accordingly.



**Fig. 5.** Tracking Algorithm.

Additionally, our system identifies unconfirmed tracks, validating them by finding corresponding detections. An extra verification step assesses Kalman Filter predictions, ensuring robustness. The system operates within a predefined zone, with vehicles exiting this zone no longer tracked. The system’s trajectory modeling aids in tracking incidents like unauthorized stops, providing valuable insights into traffic flow and congestion.

Furthermore, our system verifies a vehicle’s presence on the road and determines its movement direction, offering insights into traffic dynamics. By analyzing detected vehicles and their bounding box sizes, our system assesses road congestion, vehicle flow, and busiest routes over specific periods, enabling comprehensive traffic analysis.

**Botsort Algorithm Optimization:** To achieve superior real-time vehicle tracking, we have fine-tuned the BoT-SORT algorithm, optimizing its capabilities for our specific requirements. This enhanced version incorporates object tracking techniques, allowing continuous monitoring of vehicles within the camera’s field of view. Through our code optimization in C++ and the utilization of powerful hardware, such as the RTX 3080 GPU, our system efficiently processes up to three simultaneous video streams.

In our modification, we have devised a novel approach to combine Intersection over Union (IoU) matrices and cosine distance matrices, integrating information about both movement and appearance. Candidates with low cosine similarity or extensive IoU distances are promptly disregarded. The final cost matrix is derived from the minimum values of these matrices, facilitating accurate detections.

This adapted BoT-SORT method has been meticulously tested and proved its efficacy. In comparative tests against various trackers, our modified BoT-SORT method outperforms others in crucial metrics. Notably, it achieves remarkable accuracy and efficiency in tracking scenarios, making it a pivotal component of our tracking algorithm.

The association of tracks with new detections is determined using equation (1):

$$\text{If } (d_{i,j}^{cos} < \Theta_{emb}) \wedge (d_{i,j}^{iou} < \Theta_{iou}), \text{ then } d_{i,j}^{cos} = 0.5, \text{ else } d_{i,j}^{cos} = 1. \tag{1}$$

The cost matrix element,  $C_{ij}$ , is calculated as the minimum of cosine distance ( $d_{i,j}^{cos}$ ) and IoU distance ( $d_{i,j}^{iou}$ ):

$$C_{ij} = \min(d_{i,j}^{cos}, d_{i,j}^{iou}) \tag{2}$$

Here,  $d_{i,j}^{iou}$  represents the IoU distance between the Kalman-filtered bounding box and the bounding box of the new detection, and  $d_{i,j}^{cos}$  represents the cosine distance between the mean appearance descriptor and the new detection’s descriptor.  $\Theta_{iou}$  and  $\Theta_{emb}$  are threshold values used to filter out unlikely detections.

The Hungarian algorithm is employed to optimize the cost matrix  $C$  for the final association of existing tracks with new detections.

According to the results presented in Table 1 of the referenced article [12], this method achieved the top rank among all MOT20 trackers [13] when evaluated on the MOT16 test dataset [14]. Both MOT16 and MOT20 are widely used datasets for multi-object tracking experiments.

**Table 2.** Performance Comparison of Different Models.

Model	MOTA	DF1	HOTA	FP	FN	IDs	FPS
<i>StrongSORT++ [15]</i>	79.6	79.5	64.4	27876	86205	1194	7.1
<i>ByteTrack [16]</i>	80.3	77.3	63.1	25491	83721	2196	29.6
<i>BoT-SORT [2]</i>	<b>80.6</b>	79.5	64.6	22524	85398	1257	6.6
<i>BoT-SORT-ReID [2]</i>	80.5	80.2	65.0	22521	86037	1212	4.5

## 2.2 Road perspective and zoning

Auto zoning and Road Perspective: To accurately reconstruct the perspective of the road and create zones for speed analysis, we employ the Vanishing Point Principle from projective geometry. In this article [17] the author extracts the information of the road from driver point of view using the vanishing point principle. But we need this information from different point of view of the road. This is a similar approach [18] for road zone tracking here as well, which performs the tracking in a selected zone. We need 4 points to indicate the road boarders, two of them can be found using traffic cone detection algorithm, the operator selects two other random points lying on the other side of the road. With this given algorithm we find the vanishing point of two lines represented by their endpoints. The vanishing point is a point in a perspective image where parallel lines appear to converge. In this algorithm, the vanishing point coordinates (px, py) are calculated using algorithm 1:

---

**Algorithm 1** Find Vanishing Point

---

**function** FINDVANISHINGPOINT(line1, line2)

$((x_1, y_1), (x_2, y_2)) \leftarrow \text{line1}$

$((x_3, y_3), (x_4, y_4)) \leftarrow \text{line2}$

$\text{denominator} \leftarrow (x_1 - x_2) \times (y_3 - y_4) \times (y_1 - y_2) \times (x_3 - x_4)$

**if** denominator = 0 **then**  
end

**return** None {Lines are parallel or coincident}

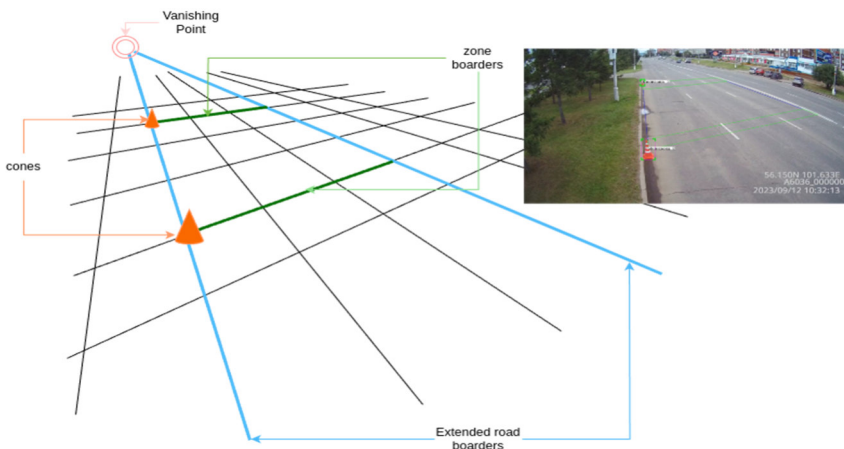
$$px \leftarrow \frac{((x_1 \times y_2 - y_1 \times x_2) \times (x_3 - x_4) - (x_1 - x_2) \times (x_3 \times y_4 - y_3 \times x_4))}{\text{denominator}}$$

$$py \leftarrow \frac{((x_1 \times y_2 - y_1 \times x_2) \times (y_3 - y_4) - (y_1 - y_2) \times (x_3 \times y_4 - y_3 \times x_4))}{\text{denominator}}$$

**return** ([px], [py])

**end function**

---



**Fig. 6.** Road Perspective and Properties.

With integrating these 4 points we construct the first zone. Analogously, by adding two other points, the algorithm would reconstruct the second zone which can be used for a different direction of the road.

Once the vanishing point has been determined, the next step is to calculate the distance of both cones to the opposite road border. Additionally, the distance of each cone to the vanishing point can be computed using the algorithm described in algorithm 2. Furthermore, the angle between the two road borders must be calculated. Ultimately, points on the road boundary can be identified using algorithm 3.

---

**Algorithm 2** Calculate Distances to Vanishing Point

---

```

Input: vanishing_point = (vx, vy), points
Output: Sorted list of distances to the vanishing point
distances ← [] for point in point do
    px, py ← [] for point distance
        ←  $\sqrt{(px - vx)^2 + (py - vy)^2}$  distances.append(distance)
end
sorted_distances ←
Sort(distances)
return sorted_distances
    
```

---

**Algorithm 3** Find Point on Line

---

```

Input: VP = (xfixed, yfixed), slope, distance_to_point, coeff
Output: Point on Line = (xnew, ynew)
xnew ← xfixed + coeff ×  $\left(\frac{\text{distance to point}}{\sqrt{1 + \text{slope}^2}}\right)$ 
ynew ← yfixed + (slope × (xnew - xfixed))
Return (xnew, ynew)
    
```

---

A crucial aspect of the zones is the inclusion of minizones (Figure 7), which constitute small sections at the beginning and end of the zones. Every vehicle must pass through these minizones to be considered in our tracking algorithm. This design eliminates vehicles that might otherwise appear solely in the middle of the zones, ensuring accurate tracking results.



**Fig. 7.** Mini Zones.

**Zone-Based Speed Calculation:** Utilizing the defined zones, we employed following mathematical algorithms to calculate vehicle speeds. By measuring the time taken by a vehicle to traverse the specified zone, combined with the known length of the zone, we can calculate the vehicle's speed.

$$t = \frac{n}{fps} \tag{3}$$

This equation is used to calculate the time ( $t$ ) in seconds given the number of frames ( $n$ ) and frames per second (fps).

$$v = \frac{x}{t} \tag{4}$$

This equation is used to calculate the velocity ( $v$ ) in meters per second ( $m/s$ ) given the distance between the cones ( $x$ ) in meters and time ( $t$ ) in seconds.

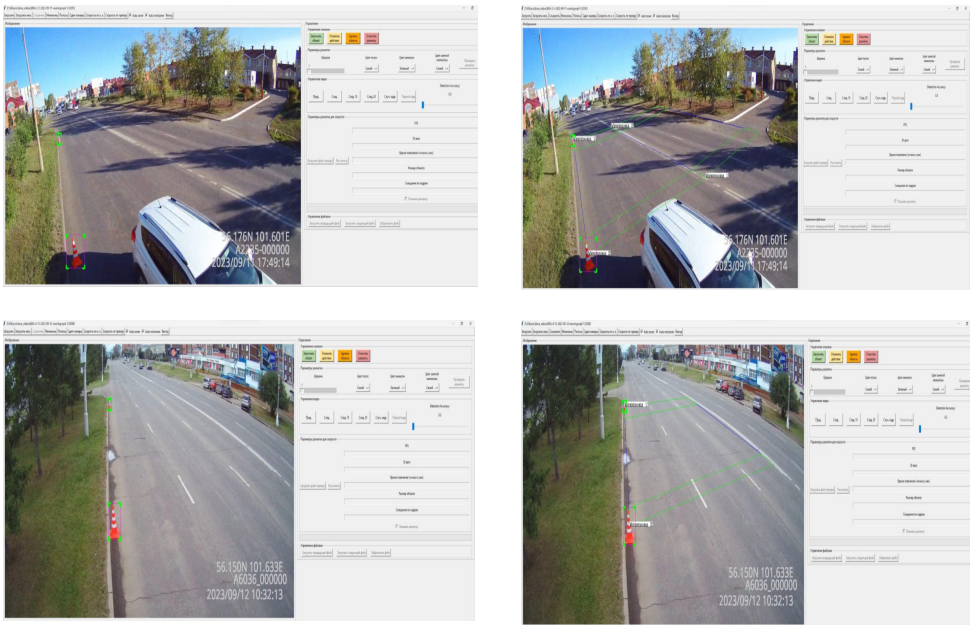
$$v_{km/h} = v \times 3.6 \tag{5}$$

And by here convert the velocity from meters per second to kilometers per hour ( $km/h$ ).

This zone-based approach offers a nuanced understanding of vehicle speeds across different sections of the road, providing valuable insights into driving behaviors.

### 3 User-Interface

We have redesigned an new user interface for the purpose of zoning the videos which supports any type of video as input and traffic cones can be detected on it and the operator can simply specify the road borders. Then the zones will be created automatically and we can save them as normalized polygons in text files. Those information about the zones and mini zones will be fed into speed calculation algorithm for further process.



**Fig. 8.** User Interface.

## 4 Results and discussion

The selection of the optimal YOLOv7 architecture, coupled with TensorRT optimization and int 16 quantization [19], ensures real-time processing of vehicular movements on 3 simultaneous video flows on RTX 3080 GPU, 10GB VRAM. The detailed classification of vehicles into seven distinct types serves as a valuable resource for ecological studies and urban planning. The tracking algorithm’s effectiveness, combined with road zoning techniques, provides nuanced insights into driving behaviors, traffic dynamics and ecological aspects. In conclusion, our research represents an advancement in understanding vehicular dynamics and their ecological impact. By bridging computer vision, mathematics, and ecology, our methodology opens new perspectives for research and applications in various fields, contributing to the creation of smarter, more sustainable cities.

**Table 3.** YOLO Detection Performance Metrics on Validation.

Model	mAP_0.5:0.95	mAP_0.5	Precision	Recall
YOLOv5m-1280	0.5431	0.8178	0.8398	0.6877
YOLOv5l-1280	0.573	0.8453	0.8677	0.7106
YOLOv5x-1280	0.5868	0.8709	0.8536	0.7561
YOLOv5x-1280	0.5787	0.8711	0.8177	0.7925
<b>YOLOv7-e6e-1280</b>	<b>0.6788</b>	<b>0.9126</b>	<b>0.9182</b>	<b>0.8541</b>
YOLOv7-tiny-640	0.5766	0.812	0.8479	0.7569
YOLOv7-tiny-1280	0.6397	0.884	0.9063	0.8049
YOLOv7-640	0.6305	0.8891	0.8973	0.818

Based on the results in Table 3 and Figure 4, the YOLOv7-e6e-1280 architecture was chosen as the detector.

A comparative table of frame processing times based on the selected architecture, framework, and precision is provided in Table 4.

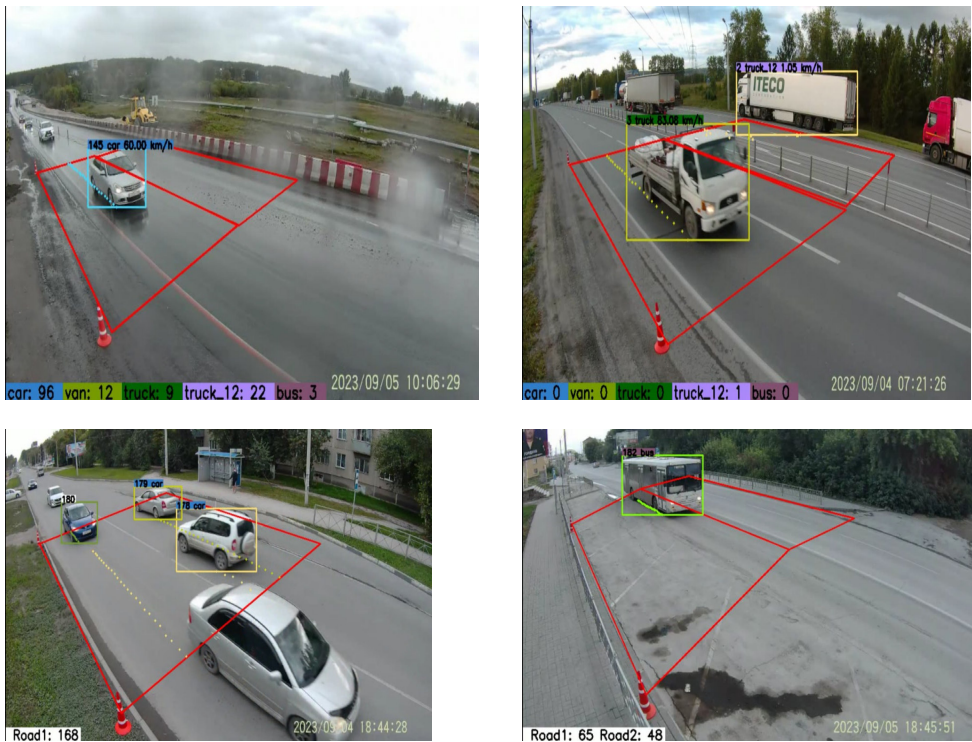
**Table 4.** Comparison of Model Processing Speeds.

Model	Framework	Precision	Time (ms) (1 img)
YOLOv7	PyTorch	fp32	24.68
		fp16	14.94
	TensorRT	fp32	16.66
		fp16	5.72
		int8	4.11
YOLOv7-tiny	PyTorch	fp32	9.80
		fp16	10.65
	TensorRT	fp32	4.83
		fp16	2.61
		int8	2.30
YOLOv7-e6e	PyTorch	fp32	43.90
		fp16	28.44
	TensorRT	fp32	35.76
		fp16	11.21
		int8	7.15

From Table 4, it is evident that architectures using the TensorRT framework process images in less time with indistinguishable precision. Reducing precision increases image processing speed. The YOLOv7-tiny architecture using TensorRT achieves the best processing speed, although it lags behind the other architectures in metrics (Table 1).

Based on the conducted research, YOLOv7 weights optimized with TensorRT and precision fp16 were selected as the solution for the detection task. These weights offer maximum processing speed with acceptable accuracy.

In Figure 9 you can see some final results of the procedure of detection, classification, tracking and speed estimation on different roads with having both directed roads



**Fig. 9.** Detection, Classification, Tracking and Speed Estimation Results.

## 5 Conclusion

Our methodology's strength lies in its ability to accurately classify vehicles into seven distinct types, ranging from cars and vans to buses and motorbikes. This detailed classification not only facilitates immediate analysis but also lays the foundation for in-depth ecological studies. Understanding the composition of vehicles on roads is essential for assessing their ecological impact and devising sustainable transportation strategies.

The tracking algorithm's effectiveness in differentiating low-confidence and high-confidence detections, coupled with the utilization of the Hungarian algorithm, ensures robust tracking of vehicles within predefined zones. This capability provides insights into traffic dynamics, enabling the identification of traffic congestion, vehicle flow patterns, and busy routes. Furthermore, the tracking system's trajectory modeling is invaluable for detecting incidents such as unauthorized stops, aiding traffic flow analysis.

This new approach for road zoning, significantly reduces the time required for accurate road geometry reconstruction. This method, combined with zone-based speed calculation, offers nuanced insights into driving behaviors across different sections of the road. By understanding vehicle speeds within specific zones, transportation authorities can implement targeted measures to enhance road safety and optimize traffic flow.

Our methodology's applications extend beyond immediate traffic analysis. The detailed data collected, including vehicle types, speeds, and movement patterns, serves as a valuable resource for urban planners, researchers, and policymakers. It enables evidence-based decision-making for sustainable urban development, more accurate CO<sub>2</sub> emission estimation, leading to reduced congestion, improved air quality, and enhanced overall quality of life for residents.

## References

1. MPreTrain Contributors. Openmmlab's pre-training toolbox and benchmark (2023). <https://github.com/open-mmlab/mmpretrain>.
2. Nir Aharon, Roy Orfaig, Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *ArXiv*, abs/2206.14651 (2022)
3. Frank Dellaert, Chuck Thorpe. *Robust car tracking using kalman filtering and bayesian templates*, In Conference on intelligent transportation systems **1** (1997)
4. Min Zhu, Yisha Liu, Yan Zhuang, Huosheng Hu, IFAC Proceedings Volumes **47(3)**, 11898-11903 (2014)
5. Hui Kong, Jean-Yves Audibert, Jean Ponce, *Vanishing point detection for road detection*, In 2009 IEEE conference on computer vision and pattern recognition, pp. 96-103 (2009)
6. Neethu John, B Anusha, Krishnan Kutty, *Procedia Computer Science* **58**, 2-9 (2015)
7. Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao, *Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7464-7475 (2022)
8. Wang Hao and Song Zhili, *Journal of Physics: Conference Series* **1684**, 012094 (2020)
9. Lingxiao He, Xingyu Liao, Wu Liu, Xinchun Liu, Peng Cheng, Tao Mei, Fastreid: A pytorch toolbox for general instance re-identification. *arXiv preprint arXiv:2006.02631* (2020)
10. Seyed Hamid Rezaatofghi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, Silvio Savarese, *Generalized intersection over union: A metric and a loss for bounding box regression*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 658-666 (2019)
11. Harold W. Kuhn, The hungarian method for the assignment problem. *Naval Research Logistics (NRL)* **52** (1955)
12. Hao Luo, Wei Jiang, Youzhi Gu, Fuxu Liu, Xingyu Liao, Shenqi Lai, and Jianyang Gu, *IEEE Transactions on Multimedia* **22**, 2597-2609 (2019)
13. Patrick Dendorfer, Hamid Rezaatofghi, Anton Milan, Javen Qinfeng Shi, Daniel Cremers, Ian D. Reid, Stefan Roth, Konrad Schindler, Laura Leal-Taix'e. Mot20: A benchmark for multi object tracking in crowded scenes. *ArXiv*, abs/2003.09003 (2020)
14. Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, Konrad Schindler. Mot16: A benchmark for multi-object tracking. *ArXiv*, abs/1603.00831 (2016)

15. Yunhao Du, Yang Song, Bo Yang, Yanyun Zhao. Strongsort: Make deepsort great again. ArXiv, abs/2202.13514 (2022)
16. Yifu Zhang, Pei Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In European Conference on Computer Vision (2021)
17. Umar Ozgunalp, Rui Fan, Xiao Ai, Naim Dahnoun, IEEE Transactions on Intelligent Transportation Systems **18**, 621-632 (2017)
18. David Beymer, Philip McLauchlan, Benjamin Coifman, Jitendra Malik, *A real-time computer vision system for measuring traffic parameters*, In Proceedings of IEEE computer society conference on computer vision and pattern recognition, pp. 495-501 (1997)
19. Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, Dmitry Kalenichenko, *Quantization and training of neural networks for efficient integer-arithmetic-only inference*, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2704-2713 (2018)