Mechatronics and Intelligent Transportation Systems
https://www.acadlore.com/journals/MITS

Enhancing Urban Traffic Management through YOLOv5 and DeepSORT Algorithms within Digital Twin Frameworks

Haoyuan Kan[1*], Chang Li[1], Ziqi Wang[2]

[1] School of Management and Engineering, Capital University of Economics and Business, 1000070 Beijing, China
[2] Faculty of Engineering and Information Technology, The University of Melbourne, 1446535 Melbourne, Australia

[*] Correspondence: Haoyuan Kan (22023210030@cueb.edu.cn)

Received: 01-16-2024 **Revised:** 02-23-2024 **Accepted:** 03-02-2024

Citation: H. Y. Kan, C. Li, and Z. Q. Wang, "Enhancing urban traffic management through YOLOv5 and DeepSORT algorithms within digital twin frameworks," *Mechatron. Intell Transp. Syst.*, vol. 3, no. 1, pp. 39–54, 2024. https://doi.org/10.56578/mits030104.

© 2024 by the authors. Published by Acadlore Publishing Services Limited, Hong Kong. This article is available for free download and can be reused and cited, provided that the original published version is credited, under the CC BY 4.0 license.

Abstract: The acceleration of urbanization and the consequent increase in population have exacerbated urban road traffic issues, such as congestion, frequent accidents, and vehicle violations, posing significant challenges to urban development. Traditional manual traffic management methods are proving inadequate in meeting the demands of rapidly evolving urban environments, necessitating an enhancement in the intelligence level of urban road traffic management systems. Recent advancements in computer vision and deep learning technologies have highlighted the potential of image processing and machine learning-based traffic management systems. In this context, the application of object detection and tracking technologies, particularly the YOLOv5 and Deep learning-based Simple Online and Realtime Tracking (DeepSORT) algorithms, has emerged as a pivotal approach for the intelligent management of urban traffic. This study employs these advanced object detection and tracking technologies to identify, classify, track, and measure vehicles on the road through video analysis, thereby providing robust support for urban traffic management decisions and planning. Utilizing digital twin technology, a virtual replica of traffic flow is constructed from camera data, serving as the dataset for training different YOLOv5 algorithm variants (YOLOv5s, YOLOv5m, and YOLOv5l). Upon comparison of training outcomes, the YOLOv5s model is selected for vehicle detection and recognition in video feeds. Subsequently, the DeepSORT algorithm is applied for vehicle tracking and matching, facilitating the calculation of vehicles' average speed based on tracking data and the temporal interval between adjacent frames. Results, stored in Comma-Separated Value (CSV) format for future analysis, indicate that the system is capable of accurately identifying, tracking, and computing the average speed of vehicles across various traffic scenarios, thereby significantly supporting urban traffic management and advancing the intelligent development of urban road traffic. This approach underscores the critical role of integrating cutting-edge object detection and tracking technologies with digital twin models in enhancing urban traffic management systems.

Keywords: Urban traffic management; Deep learning; YOLOv5; Deep simple online and realtime tracking (DeepSORT); Digital twins

1 Introduction

With the rapid development of the economy and the rapid growth of per capita GDP, the speed of urbanization is also very fast. At present, people's living standards are improving, and private cars are also increasing year by year. Due to the convenience brought by their travel, the number of cars in use is increasing exponentially. Taking China as an example, by the end of February 2023, the total number of motor vehicles in China had reached 319.03 million [1]. The mismatch between the increase in local demands for urban transportation and the disproportionate growth of local transportation supplies may cause serious traffic congestion issues [2]. Due to the complex road environment, various external environmental factors, and human factors, traffic accidents occur frequently [3], and every time a traffic accident occurs, it will cause congestion on one side of the road. Traffic congestion has become one of the biggest obstacles restricting city economic development; it results in high consumption of fuel, increases the cost of commutes, and also pollutes the environment [4].

Many corresponding methods, technologies, and means have emerged to address the current influencing factors of traffic regulation. It can perform traffic flow statistics, prediction, vehicle trajectory tracking, vehicle speed

measurement, and dispatch traffic police patrols to take appropriate measures in a timely manner to solve road traffic problems when they are discovered. The success of an ITS is mainly determined by the quality of traffic information provided to traffic stakeholders and the ability to apply traffic information towards developing policies, control systems, and traffic prediction models [5]. Therefore, knowing how to accurately and effectively obtain the types and quantities of motor vehicles on the road and provide higher-quality data is a prerequisite for determining the effectiveness of intelligent and intelligent management of road traffic.

This article mainly integrates YOLOv5 and DeepSORT algorithms to achieve vehicle recognition and speed measurement functions on datasets generated through digital twin technology, thereby effectively improving traffic management efficiency. Among them, YOLOv5 is a deep learning model used for object detection tasks that can quickly and accurately detect various objects, such as vehicles and pedestrians in traffic scenes. DeepSORT is a multi-target tracking algorithm that can track and classify vehicles based on detection and calculate key data, such as vehicle speed, in real-time.

Firstly, by combining the above algorithms, traffic conditions can be monitored, effectively reducing the occurrence of traffic congestion. Secondly, through the speed measurement function, the driving speed of vehicles on the road can be detected, and speeding vehicles can be reminded and punished, thereby enhancing road traffic safety. Finally, historical traffic data can be saved to provide support for urban planning and public transportation decision-making.

This article will elaborate on the principles of YOLOv5 and DeepSORT algorithms and compare the different variants of YOLOv5. At the same time, the implementation process and display of experimental results are aimed at providing useful references for research and application in the field of urban road traffic management.

The YOLOv5 and DeepSORT algorithms are widely used, and the urban road traffic management system based on YOLOv5 and DeepSORT has attracted widespread attention from researchers. Payne et al. proposed a smart traffic management system for urban roads called “Smart Roads”, which uses YOLOv5 and DeepSORT algorithms for vehicle detection and tracking and analyzes historical data for traffic condition prediction and signal control strategy optimization. At the same time, the system can also achieve functions such as overspeed detection and traffic flow monitoring. In addition, Huang et al. also developed a traffic flow monitoring system based on deep learning, which uses YOLOv5 and DeepSORT algorithms to detect and track vehicles and calculates real-time traffic flow data such as vehicle speed and density. In the field of smart traffic management on urban roads, YOLOv5 and DeepSORT algorithms based on the Python programming language have become hot research directions. Related studies have shown that the application of these technologies can effectively improve traffic management efficiency, alleviate traffic congestion problems, and enhance road traffic safety, which is an important direction for the future development of urban road traffic.

In order to effectively improve traffic management efficiency, alleviate traffic congestion, and enhance road traffic safety, this paper studies the detector YOLOv5 and the tracking algorithm DeepSORT and develops a high-performance real-time multi-target detection and tracking speed measurement model. The main content of the paper is as follows:

(1) Vehicle detection based on the YOLOv5 model. In vehicle recognition scenarios, the YOLO series algorithm performs the best. This article mainly introduces the development process and main improvement points of the YOLO series object detection algorithms and discusses in detail the backbone network structure and operation methods of these algorithms. We have provided a brief introduction to the YOLOv1, YOLOv2, YOLO9000, YOLOv3, and YOLOv4 algorithms, with a focus on analyzing the network structure of the YOLOv5 algorithm and the operational principles of each constituent module. The YOLO series algorithm performs better. Through investigation, it has been shown that the YOLO series algorithm performs better in the field of vehicle recognition and detection. This paper compares YOLO series algorithms and different variants of YOLOv5, and finally uses YOLO5’s model in the YOLOv5 algorithm for vehicle recognition detection.

(2) Vehicle tracking based on the DeepSORT algorithm. This article adopts the DeepSORT algorithm for vehicle tracking, which is a vehicle tracking method that combines motion models and appearance feature models. Its predecessor was the SORT algorithm, which is highly effective in solving multi-objective tracking problems.

(3) Vehicle speed measurement. This article designs an algorithm to measure vehicle speed. The algorithm first needs to convert the position difference (i.e., pixel distance) between the current frame and the previous frame of the vehicle into the true spatial distance. Meanwhile, due to the possibility that the video shooting angle may not be perpendicular to the vehicle’s movement trajectory, the calculated speed value may be lower than the actual speed. Therefore, it is necessary to calculate the actual speed of the vehicle based on the pixel width of the vehicle category, the video frame rate, and the mapping relationship from pixel distance to real spatial distance. Store the calculated vehicle ID and speed in a CSV file for subsequent queries and analysis.

The rest of this article is organized as follows: Section 2 analyzed the YOLO series algorithm and different variants of YOLOv5 and selected the YOLOv5s model for vehicle detection. Section 3 analyzed the principle of the DeepSORT algorithm and used it for vehicle tracking. Section 4 designed a speed measurement algorithm to

measure the speed of vehicles in the video, process errors, and output the vehicle ID and speed to a CSV file for subsequent queries and analysis. Section 5 provides a summary and identifies future research directions.

2 YOLO Series Vehicle Detection Algorithms

2.1 YOLO Algorithms Throughout History

The YOLO network is trained by data sets, and the model tests traffic flow statistics. The statistical results reflect the effect of traffic flow monitoring [6]. With the development of the YOLO algorithm, detection performance has been rapidly improved.

The first-generation YOLO algorithm in the YOLO series was proposed by Redmon et al. [7]. This method first proposed an anchor box-based vehicle detection method, inspired by the Google Net network, and built an end-to-end object detection framework using the VGG-16 convolutional neural network (CNN). However, YOLOv1 has issues such as inaccurate positioning and poor detection performance for small objects.

In 2017, Redmon and Farhadi [8] proposed YOLO9000, the second-generation algorithm of the YOLO series, YOLOv2. Compared with YOLOv1, YOLOv2 has improved its network structure by using a deeper CNN (Darknet-19), as shown in Table 1. The introduction of anchor boxes and batch normalization technology significantly improves detection performance and speed.

Table 1. Darknet-19

Type	Filters	Size/Stride	Output
Convolutional	32	3*3	224*224
Maxpool		2*2/2	112*112
Convolutional	64	3*3	112*112
Maxpool		2*2/2	56*56
Convolutional	128	3*3	56*56
Convolutional	64	1*1	56*56
Convolutional	128	3*3	56*56
Maxpool		2*2/2	28*28
Convolutional	256	3*3	28*28
Convolutional	128	1*1	28*28
Convolutional	256	3*3	28*28
Maxpool		2*2/2	14*14
Convolutional	512	3*3	14*14
Convolutional	256	1*1	14*14
Convolutional	512	3*3	14*14
Convolutional	256	1*1	14*14
Convolutional	512	3*3	14*14
Maxpool		2*2/2	7*7
Convolutional	1024	3*3	7*7
Convolutional	512	1*1	7*7
Convolutional	1024	3*3	7*7
Convolutional	512	1*1	7*7
Convolutional	1024	3*3	7*7
Convolutional	1000	1*1	7*7
Avgpool		Global	1000

The YOLOv3 algorithm was proposed by Redmon and Farhadi [9] in 2018. YOLOv3 added modules such as residual network (ResNet) and feature pyramid network (FPN) to the network structure, further improving detection accuracy and achieving the best results in the field of object detection at that time. YOLOv3 can achieve good performance on a powerful GPU, but the training process is still very time-consuming. Under the conditions of embedded computing devices with limited computing power and limited memory hardware, using this method to perform real-time object detection is still very difficult [10].

In April 2020, Bochkovskiy et al. [11] made significant improvements and optimizations on the basis of YOLOv3 and proposed the YOLOv4 algorithm. It introduces innovative designs such as the CSPDarknet53 network structure, SPP module, and SAM module, further enhancing detection accuracy and speed. In addition, YOLOv4 adopts strategies such as multi-scale data augmentation and mosaic data augmentation during the training process, increasing the diversity and richness of the dataset and thereby making the detection effect more excellent.

2.2 YOLOv5 Algorithm Introduction

YOLOv5 was released in June 2020 [12]. Compared with previous versions, YOLOv5 has made improvements and optimizations in network structure, feature extractor, and training strategy, achieving more efficient object detection. This section mainly elaborates on YOLOv5 from various aspects, such as data augmentation, network design principles, and model loss functions.

2.2.1 Data enhancement

The YOLOV5 model mainly uses the Mosaic method [13] for data augmentation at the input end. The principle of its method is similar to the CutMix method [14], but it is different from Mosaic in that it randomly takes four images to stitch together for training in order to reduce the use of GPUs and effectively improve the detection and localization ability of small targets. In addition, YOLOv5 also uses an adaptive anchor box training method [15], which embeds the calculation of anchor boxes into the overall code, generates prediction boxes based on the initial fixed size, and then continuously updates the calculation to obtain the optimal anchor box size to adapt to different datasets. At the input end, YOLOv5 uses adaptive scaling to solve the problem of the input image's aspect ratio not being fixed. It calculates the scaling ratio and size, scales the input image appropriately, and fills the smaller side with black edges after scaling to ensure that the scaled size is a multiple of 32, greatly improving the algorithm's inference speed.

2.2.2 Network structure

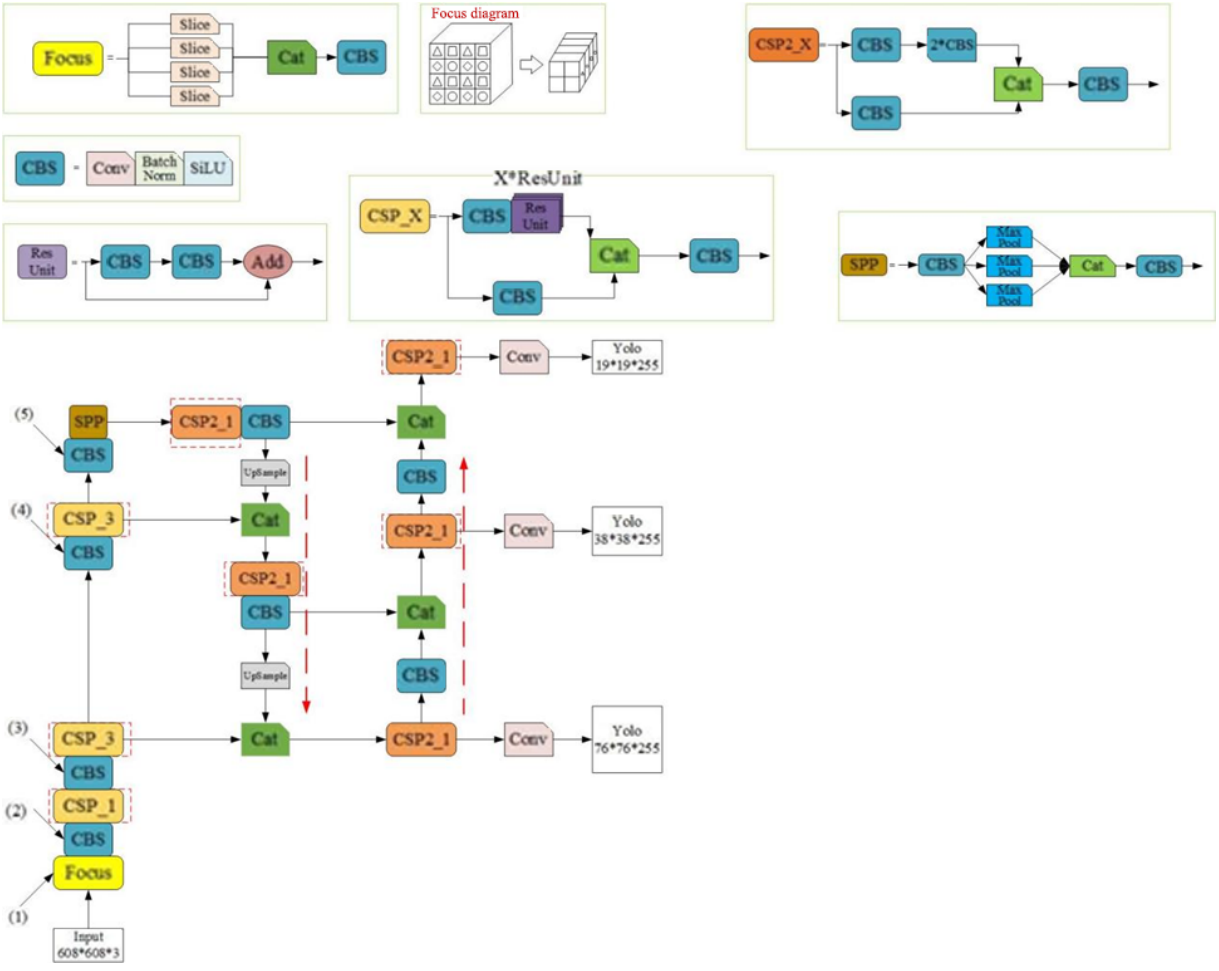


Figure 1. YOLOv5 network structure

YOLOv5 adopts a new network structure, namely Cross Stage Partial Network (CSPNet). YOLOv5 also introduces some new technologies, including the feature pyramid, Path Aggregation Network (PANet), and channel attention mechanism, to improve detection accuracy. At the same time, in order to solve the problem of data imbalance, YOLOv5 adopts a class-balanced cross-entropy loss function and uses an adaptive learning rate adjustment method to improve the model's generalization ability. In addition, YOLOv5 also supports training strategies such as multi-scale prediction and image flipping data augmentation to improve the robustness and generalization ability of the model.

Compared with other deep learning algorithms, YOLOv5 has a smaller number of model parameters and can run under lower computational resources and memory constraints. YOLOv5 has different variants, and the algorithm flow for each variant is shown in Figure 1, Tables 2 and 3.

Table 2. YOLOv5 depth (red dashed box)

	YOLOv5s	**YOLOv5m**	**YOLOv5l**	**YOLOv5x**
First CSP	CSP_1	CSP_2	CSP_3	CSP_4
Second CSP	CSP_3	CSP_6	CSP_9	CSP_12
Third CSP	CSP_3	CSP_6	CSP_9	CSP_12
First CSP2	CSP2_1	CSP2_2	CSP2_3	CSP2_4
Second CSP2	CSP2_1	CSP2_2	CSP2_3	CSP2_4
Third CSP2	CSP2_1	CSP2_2	CSP2_3	CSP2_4
Forth CSP2	CSP2_1	CSP2_2	CSP2_3	CSP2_4
Fifth CSP2	CSP2_1	CSP2_3	CSP2_3	CSP2_4

Table 3. YOLOv5 width (black arrow)

Number of Convolution Kernels	**YOLOv5s**	**YOLOv5m**	**YOLOv5l**	**YOLOv5x**
(1)	32	48	64	80
(2)	64	96	128	160
(3)	128	192	256	320
(4)	256	384	512	640
(5)	512	768	1024	1280

2.2.3 Model loss function

In the training process of object detection, the common phenomenon of imbalanced samples has a significant impact on the performance of the model, so a loss function is usually used to calculate the model loss. In the model bounding box regression loss, the IoU intersection union ratio is usually used as a measure. Assuming A is the predicted box and B is the actual box, its calculation is shown in Eq. (1).

$$IoU\{A,B\} = \frac{\text{area}(A) \cap \text{area}(B)}{\text{area}(A) \cup \text{area}(B)} \tag{1}$$

In general, the closer the value of IoU is to 1, the greater the degree of matching, and the smaller the regression loss. However, when using IoU to measure losses, it may occur that the predicted box and the actual box do not overlap, resulting in $IoU_{Loss} = 1$. Or there may be situations where the predicted boxes are the same and the calculated IoU is equal, with IoU being equal, and IoU_{Loss} is unable to distinguish the phenomenon of two intersecting. To address this issue, the $GIoU_{Loss}$ function was adopted. $GIoU_{Loss}$ has added a measurement method for intersection scale. Firstly, the minimum box area A_C containing both A and B target boxes and the intersection area U of the$A$ and B target boxes are calculated. The specific calculation is shown in Eq. (2).

$$\begin{cases} GIoU = IoU - \frac{|A_C - U|}{|A_C|} \\ GIoU_{Loss} = 1 - GIoU \end{cases} \tag{2}$$

But if the predicted box obtained happens to be inside the target box and has the same area size, the difference size obtained through $A_C - U$ is the same, resulting in the same $GIoU_{Loss}$ value and the inability to distinguish its position. In order to improve training stability, GIoU simultaneously calculates the overlapping area, center point distance, and aspect ratio of the target boxes A and B, as shown in Eq. (3).

$$\begin{cases} CIoU = IoU - \frac{\rho^2}{c^2} - \frac{v^2}{(1-I_0 u)+v} \\ CIoU_{Loss} = 1 - GIoU \\ v = \frac{4}{\pi}\left(\arctan\frac{x_{l2}-x_{l1}}{y_{l2}-y_{l1}} - \text{acrtan}\frac{x_{\rho 2}-x_{\rho 1}}{y_{\rho 2}-y_{\rho 1}}\right)^2 \end{cases} \tag{3}$$

where, ρ is the Euclidean distance between the center points of target boxes A and B, and c is the minimum diagonal length of the rectangle surrounding A and B. v indicates the similarity between the aspect ratios of boxes A and B. When the aspect ratios are equal, 0 is taken, and when the aspect ratio difference is infinitely large, 1 is taken.

During the object detection process, YOLOv5 uses NMS non maximum suppression to filter out excess detection boxes.

2.3 Experimental Process and Analysis

This paper chooses to use a subset of the COCO dataset, namely COCO128, to train YOLOv5, which only contains 128 categories, 7321 images, and 10340 annotated objects. This greatly reduces the training time and resource requirements of the model while also enabling faster validation of model performance and experimentation. Although the COCO128 dataset is relatively small, it can still provide models with rich images and target categories, enabling them to quickly learn the basic knowledge of object detection and lay the foundation for more complex datasets and scenes. The dataset is shown in Figure 2.

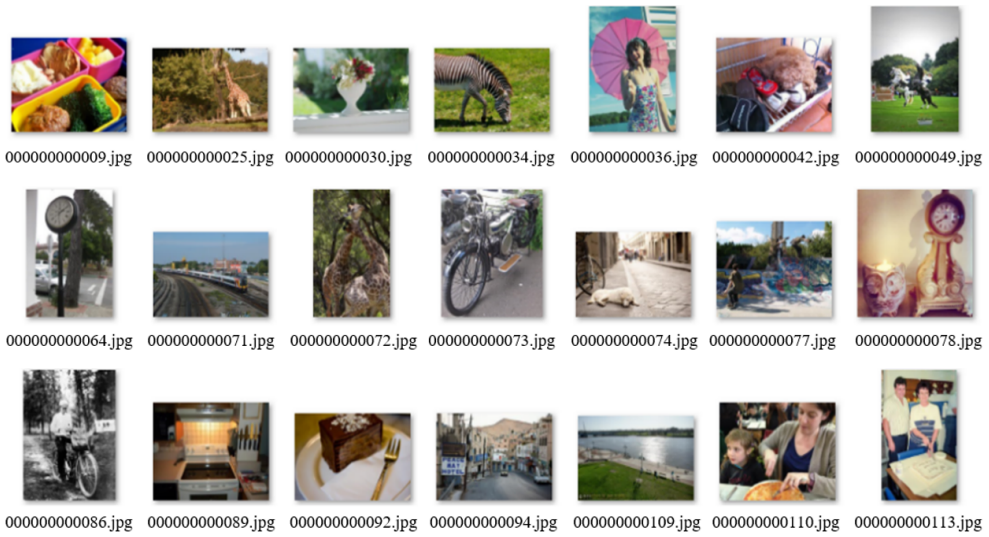


Figure 2. COCO dataset

This article evaluates the test results using values such as precision (P), recall (R), and mean average precision (MAP).

Due to the need to consider detection speed in vehicle tracking tasks, this article compared YOLOv5s, YOLOv5m, and YOLOv5l.

The momentum parameter set for training is 0.937, the weight attenuation factor is 0.0005, the initial learning rate is set to 0.01, and the IoU threshold is set to 0.2. By setting a smaller learning rate, the training process is more stable, and overfitting is prevented. The training results of the YOLOv5 model after training are shown in Table 4.

Table 4. Training results of different variants of YOLOv5

	YOLOv5s	YOLOv5m	YOLOv5l
Box (P)	0.916	0.933	0.955
Box (R)	0.898	0.925	0.934
MAP50 (B)	0.955	0.971	0.975
MAP50-95 (B)	0.773	0.84	0.873
Mask (P)	0.897	0.971	0.938
Mask (R)	0.822	0.866	0.898
MAP50 (M)	0.862	0.911	0.935
MAP50-95 (M)	0.621	0.674	0.719
Speed	6h	10h	18h

Among them, Box (P) refers to the accuracy indicator used in object detection tasks, namely the bounding box prediction accuracy (Precision);

Box (R) refers to the recall metric used in object detection tasks, namely the bounding box prediction recall (Recall);

MAP50 (B) refers to the average accuracy metric (AP) used in object detection tasks, which calculates the AP value for all categories when the Intersection over Union (IoU) between predicted and true results is greater than or equal to 0.5, and takes its average value;

MAP50-95 (B) refers to the AP metric used in object detection tasks, which calculates the AP values for all categories with an IoU greater than or equal to 0.5 to 0.95 between predicted and true results, and takes the average value; Mask (P) refers to the accuracy metric used in semantic segmentation tasks, namely pixel-level prediction accuracy (Precision);

Mask (R) refers to the recall metric used in semantic segmentation tasks, namely pixel-level predicted recall (Recall);

MAP50 (M) refers to the average accuracy metric (AP) used in semantic segmentation tasks, which calculates the AP value for all categories when the IoU between predicted and true results is greater than or equal to 0.5, and takes its average value; MAP50-95 (M) refers to the AP metric used in semantic segmentation tasks, which calculates the AP values of all categories with an IoU greater than or equal to 0.5 to 0.95 between predicted and actual results, and takes the average value. Speed refers to the training speed of Torch 1.8.0+CPU on the Windows 10 system.

The overall trend of the three models is shown in the above figure, so the best generation with test results will be displayed in the table. Among the three models, YOLOv5s has the fastest recognition speed, so in subsequent models, YOLOv5s will be used for further experiments.

When training the network model, the loss and MAP curves of the YOLOv5s algorithm obtained are shown in Figure 3. After multiple iterations of training, the entire training converges well to a relatively low level. It is evident that the loss curves of bounding box regression, target regression, and classification regression have basically stabilized, and the model performance is relatively robust. At the same time, the benchmark network has reached an MAP of around 0.97_0.5 (MAP at IoU value of 0.5) and around 0.9 MAP_0.5:0.95 (MAP with an IoU value of 0.5 to 0.95 and a step size of 0.05). From this, it can be seen that when the sample distribution is balanced, YOLOv5 detection shows less overfitting.

From Figure 4, it can be seen that the algorithm can excellently complete the target classification task, identifying cars, trucks, and motorcycles in the vehicle.

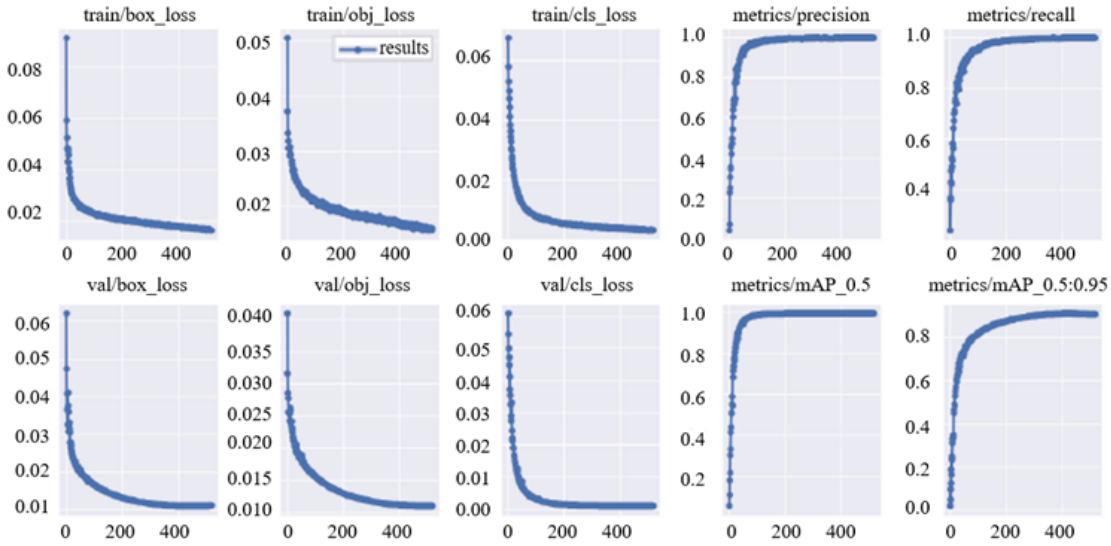


Figure 3. YOLOv5s loss function

(a) Motorcycle (b) Truck

Figure 4. Different vehicle identification

3 Vehicle Tracking Based on DeepSORT Algorithm

3.1 Analysis of DeepSORT Algorithm

SORT is a classic multi-target tracking algorithm that uses a Kalman filter to predict the position and velocity information of the target in the next frame and the Hungarian algorithm to associate the detected target in the current frame with the tracked target in the previous frame, thereby achieving multi-target tracking.

The DeepSORT algorithm is a multi-objective tracking algorithm that combines deep learning and the classic object tracking algorithm SORT. The core idea of the DeepSORT algorithm is to extract target feature vectors through deep learning and combine them with appearance models to improve the accuracy and robustness of multi-target tracking. It uses CNN to learn target features, uses cosine similarity to measure the similarity between target features between different frames, and uses greedy matching-based methods to determine target matching between different frames. Compared with the SORT algorithm, the DeepSORT algorithm has better performance and robustness in complex scenarios, such as overlapping and crossing between targets. In the competition for multi-target tracking, the DeepSORT algorithm has achieved good performance and has become one of the most popular multi-target tracking algorithms in the industry [16].

3.1.1 Hungarian algorithm

The Hungarian algorithm is an algorithm used to solve the maximum matching problem (assignment problem) in bipartite graphs [17]. A bipartite graph refers to a graph in which all nodes can be divided into two disjoint sets, and the two endpoints of all edges in the graph belong to these two sets, respectively. The maximum matching problem refers to selecting the most edges in a bipartite graph so that these edges have no common endpoints. The Hungarian algorithm in DeepSORT calculates the appearance similarity between each detected target and the previously tracked target and uses it as an additional constraint. This similarity can be obtained by calculating the Mahalanobis distance of the appearance features between two targets. At the same time, DeepSORT also performs keyframe sampling on tracked targets to further improve the accuracy and stability of target matching.

3.1.2 Kalman filtering algorithm

The Kalman filter, evolved from Bayesian filtering, can predict the state data of the target vehicle in the current frame of the video based on the state data of the target vehicle in the previous frame of the video [18]. The state data vector of the target vehicle consists of 8 variables, including the x and y coordinates of the center point of the detection box, the aspect ratio of the detection box, the height of the detection box, and the variation of these variables. The Kalman filtering algorithm consists of two processes: prediction and parameter update, and the corresponding calculation processes are Eq. (4) and Eq. (5).

$$\begin{cases} \hat{x}_k^- = A\hat{x}_{k-1} \\ P_k^- = AP_{k-1}A^T + Q \end{cases} \tag{4}$$

$$\begin{cases} K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1} \\ \hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right) \\ P_k = \left(1 - K_k H\right) P_k^- \end{cases} \tag{5}$$

In the formula, $\hat{x}_{k-1}$ is the state of the target vehicle in the k $-$ 1 video frame, $\hat{x}_k^-$ is the predicted state value of the target vehicle in the k video frame, and A and Q are both $8\times$ the matrix of 8, A is the state transition matrix, Q is the estimated noise covariance matrix, P_{k-1} represents the state estimation covariance of the target vehicle in frames k $-$ 1, and P_k^- is the state estimation covariance of the target vehicle in frames k. After the k-th prediction, it is necessary to update the parameters based on the k-th detection results. Eq. (5) describes the process of parameter updates. Where K_k represents the k-th frame Kalman gain. H is the state observation matrix. R is $4\times$ the covariance matrix of observation noise for 4. z_k represents the state observation value of the k-th frame. $\hat{x}_k$ and P_k^- are the best estimated values and error covariance of the state at frame k, respectively, and they are used for predicting the target vehicle state at frame $k + 1$ [19].

In the Kalman filtering algorithm, the error between the target vehicle prediction box and the target vehicle detection box is measured by the Mahalanobis distance, which serves as the loss function of the Kalman filtering. By using Kalman filtering, the optimal position data of the target vehicle in the current video frame can be predicted based on its state information in the previous frame. If the distance between the detection position of the target vehicle and the optimal position is less than the threshold, it is judged as the same vehicle; otherwise, it will be judged as different vehicles. The values of matrices A, Q, and R can be found in Eqs. (6)-(8), where h is the height of the detection box.

$$A_{ij}\begin{cases}1 & i=j=1,2,3,\ldots,8\\ 1 & i+j=6\\ 0 & \text{other}\end{cases} \tag{6}$$

$$Q_{ij}\begin{cases}\left(\frac{h}{20}\right)^2 & i=j=1,2,4\\ 10^{-4} & i=j=3\\ \left(\frac{h}{120}\right)^2 & i=j=5,6,8\\ 10^{-10} & i=j=7\\ 0 & \text{other}\end{cases} \tag{7}$$

$$R_{ij}\begin{cases}\left(\frac{h}{20}\right)^2 & i=j=1,2,4\\ 10^{-2} & i=j=3\\ 0 & \text{other}\end{cases} \tag{8}$$

Mahalanobis distance is an operational method for calculating the covariance distance between two data points, while in the Kalman filtering algorithm, by predicting the state of the target vehicle, the prediction box of the target vehicle can be obtained. To measure the error between the prediction box and the detection box of the target vehicle in the current frame, Mahalanobis distance can be used as the loss function. By minimizing the Mahalanobis distance, the best prediction results can be obtained, thereby achieving tracking and recognition of the target vehicle. Eq. (9) is the calculation formula for Mahalanobis distance, where x is the vector composed of the predicted box position coordinate values, μ is the mean of x, Σ is the variance of x.

$$D_m(x)=\sqrt{(x-\mu)^T\Sigma^{-1}(x-\mu)} \tag{9}$$

3.2 DeepSORT Algorithm Process

The DeepSORT algorithm is mainly divided into two stages: object detection and object tracking.

3.2.1 Object detection stage

In the object detection stage, the DeepSORT algorithm uses an object detector to detect targets in the image, and in this paper, YOLOv5 is used for detection. For each detected target, the DeepSORT algorithm uses CNN to extract its feature vectors, which will be used for subsequent target tracking.

3.2.2 Object tracking stage

In the object tracking phase, the DeepSORT algorithm uses the classic target tracking algorithm SORT to track each detected target. On the basis of the SORT algorithm, the DeepSORT algorithm introduces a CNN based appearance model and a greedy matching based association algorithm to improve the accuracy and robustness of multi-target tracking. Specifically, the tracking process of the DeepSORT algorithm is as follows:

(1) Prediction: For each detected target in the current frame, use a Kalman filter to predict it and obtain its position and velocity information for the next frame.

(2) Matching: For each predicted target, calculate the cosine similarity between its feature vector and the feature vector of the tracked target in the previous frame, and use the Hungarian algorithm to match each predicted target with its most similar target.

(3) Update: For each matched target, use a Kalman filter to update its position, velocity information, and feature vectors. For unmatched targets, consider them as new targets and track them.

(4) Delete: For some targets that have not been matched in consecutive frames, they are considered leaving targets and removed from the tracking list.

3.3 Experimental Process and Analysis

3.3.1 Improvement

DeepSORT is mainly used for pedestrian tracking, and we need to modify its input. The original input was a rectangular box with a size of 128 (height) × 64 (width). To meet the needs of vehicle tracking, it needs to be modified to a rectangular box of 64 (height) × 128 (width). This can better adapt to the shape and size of pedestrians.

Afterwards, the original kernel size was 8×4, which refers to the size of the convolutional kernel in the CNN. In order to adapt to the new rectangular box size, it needs to be modified to 4×8. This can better match the new input rectangle.

3.3.2 Experimental process

In order to train the DeepSORT model, the VERI-WILD training set was used for training. VERI-WILD is a dataset used for visual object recognition and tracking that includes 400000 images from 40000 vehicle identifiers. The data is shown in Figure 5.

Figure 5. VERI-WILD training set

The model has a loss rate of 0.18 and an accuracy of 96.2% in the training set; the loss rate of the test set is 0.65, and the accuracy is 80.2%. The result is shown in Figure 6. These results indicate that the DeepSORT model has high accuracy and reliability for vehicle tracking tasks.

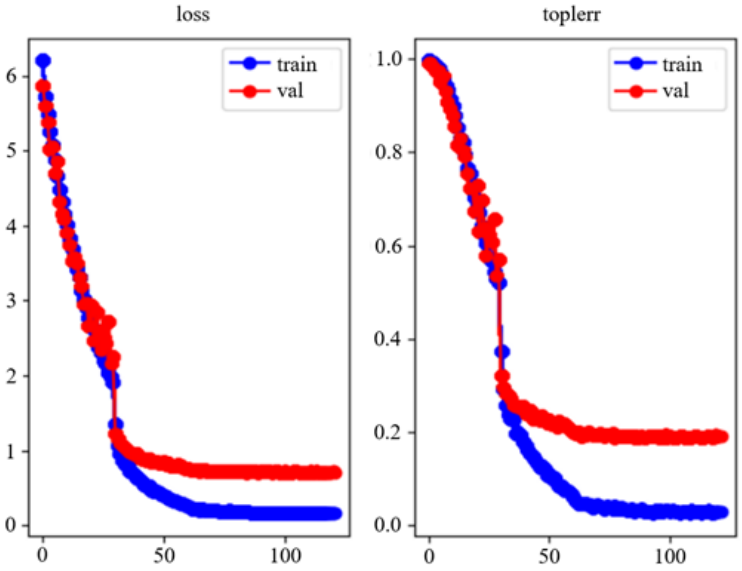
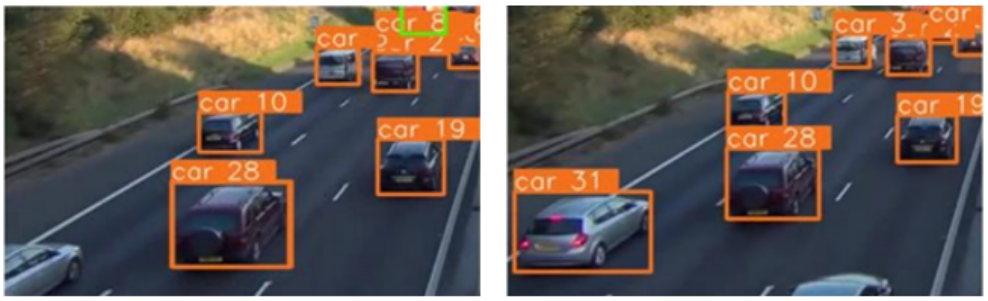


Figure 6. DeepSORT algorithm loss function

3.3.3 Analysis

The DeepSORT algorithm associates and matches the detector and tracker based on the detection results obtained from the YOLOv5s model. In this way, the target can be tracked between different frames, and continuous tracking of the target can be achieved.

Subgraph (a) of Figure 7 shows the detection result of the 35th frame of the video, and Subgraph (b) of Figure 7 shows the detection result of the 62nd frame. By observing the images in the video, we can see that the DeepSORT algorithm has successfully assigned a unique ID to each vehicle, and as the video frame moves, the annotation box for each vehicle also moves accordingly. This indicates that the algorithm is very effective in tracking vehicle targets and can stably maintain their ID information unchanged. At the same time, the algorithm can also stably follow the movement of the vehicle and maintain the stability of the annotation box. These characteristics all indicate that the DeepSORT algorithm is an efficient target tracking algorithm that can accurately track and identify vehicle targets in complex environments.



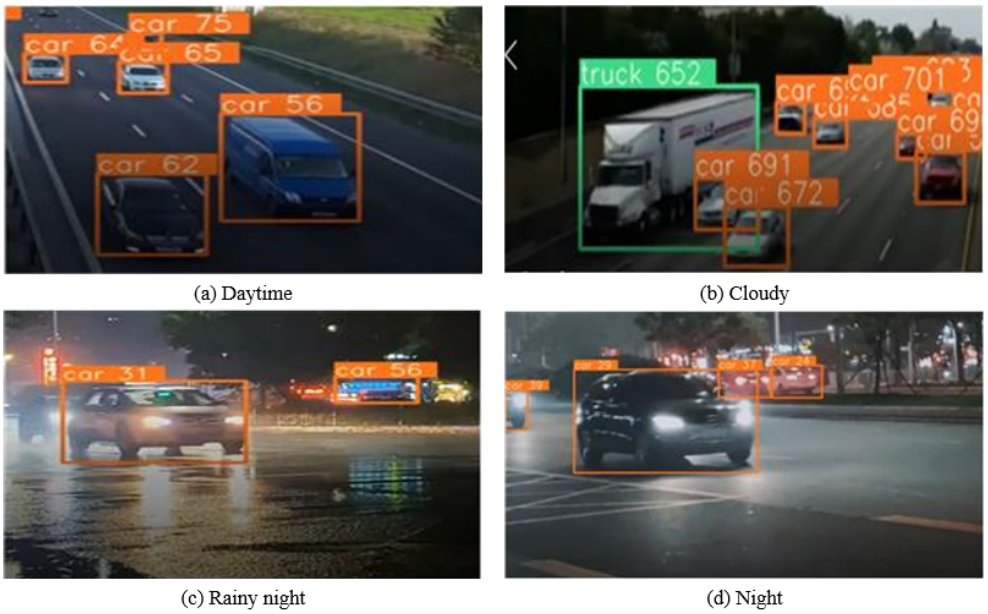
(a) Frame 35 (b) Frame 62

Figure 7. Detection result

(a) Before occlusion (b) After occlusion

Figure 8. Detection diagram

Observing photos in Figure 8, we can observe that even after being obstructed by the vehicle with ID 92, the original vehicle with ID 95 can still be correctly recognized. This indicates that the DeepSORT algorithm still has very good tracking performance when encountering dynamic camera environments. This algorithm can deal with some phenomena that may cause camera shake, such as occlusion between vehicles, so as to accurately track the target. This characteristic makes the DeepSORT algorithm a very promising target tracking algorithm with broad application prospects in practical applications [20].



(a) Daytime (b) Cloudy

(c) Rainy night (d) Night

Figure 9. Detection diagram

Observing photos in Figure 9, this paper demonstrates the tracking performance of the DeepSORT algorithm under different lighting, weather, and road conditions by selecting four different scenes: day, cloudy, rainy night,

and night for video detection and tracking. Through result analysis, it can be found that in these four scenarios, the DeepSORT algorithm can stably track car targets and accurately identify and track cars. This indicates that the algorithm has excellent accuracy and robustness, and can demonstrate excellent tracking performance in different scenarios. This algorithm can maintain a stable tracking frame and accurately identify and track targets, whether it is in low-light nights or complex weather conditions such as rainy days, and has high practical value. This tracking algorithm has a very broad application prospect in fields such as traffic management, urban planning, and public safety.

Finally, through testing multiple videos, this article selected five detection videos from the same day, but with different detection speeds, densities, and shooting angles, in order to conduct accurate testing. Among the 5 detection videos, the DeepSORT algorithm can display the number of vehicles on the current video screen and the cumulative number of vehicles in the video, respectively. Table 5 shows the counting accuracy. From the data in Table 5, it can be seen that the vehicle flow detection algorithm based on YOLOv5s and DeepSORT has good detection performance for 5 different vehicle flow videos, with an average accuracy of 93.6%, MAE of 18.2, MSE of 460.2, and RMSE of 21.5. This indicates that the algorithm proposed in this article has strong adaptability to practical situations and also has good detection accuracy in high-density or high-speed situations. This algorithm has great application value.

Table 5. Five different video detection effects for traffic flow

Traffic Flow Video	Actual Number of Vehicles	Detection Result	Accuracy
Video 1	96	82	0.85
Video 2	348	332	0.95
Video 3	1168	1130	0.97
Video 4	720	698	0.97
Video 5	150	141	0.94

4 Vehicle Speed Measurement

4.1 Design Background

Road traffic is an important component of urban transportation systems, and the speeding of vehicles is one of the main causes of traffic accidents. Therefore, in order to prevent traffic accidents, government departments need to supervise and control vehicles on the road to ensure that their driving speed complies with traffic rules and road safety standards. The vehicle speed measurement function is a technical means used to measure the speed of a vehicle. It can monitor and record the speed of the vehicle in real time, and punish vehicles that exceed the speed limit.

4.2 Algorithm Flow

This paper designs a function for vehicle speed measurement. Firstly, by traversing the locations list, obtain the vehicle IDs (present_IDs and prev_IDs) tracked in the current and previous frames, respectively, and find the valid vehicle IDs (work_IDs) detected in both frames of the image. Then, based on the vehicle ID index, store the effective location information of the detected vehicle in the current frame and the previous frame (including the center point x coordinate, center point y coordinate, and target number) in work_locations and work_prev_locations, respectively. By calculating the position difference between work_locations and work_prev_locations, we can obtain the distance the vehicle moves in pixel space.

However, due to the fact that the video shooting perspective may not be perpendicular to the vehicle's movement trajectory, pixel distance cannot be directly converted into true spatial distance. Therefore, it is necessary to calculate the actual speed of the vehicle based on the pixel width of the vehicle categories, the video frame rate, and the mapping relationship from pixel distance to real spatial distance. In the calculation process, we keep one decimal place and set the speed unit to km/h. Finally, we store the vehicle speed and its corresponding ID in the speed two-dimensional list and return the result.

4.3 Experimentation

Use the YOLOv5s model for vehicle detection, identify vehicles, and obtain vehicle position information in each frame of the image.

Use the DeepSORT algorithm to track vehicles and keep the vehicle ID of the same vehicle matched between adjacent frames unchanged.

By traversing the vehicle position information of each frame, the tracked vehicle IDs in the current and previous frames are obtained, and valid vehicle IDs detected in the two frames of the image are found. By observing the images in the video and based on the vehicle ID index, the effective location information of the detected vehicle

in the current and previous frames (including the center point x coordinate, center point y coordinate, and target number) is stored in work_locations and work_prev_locations, respectively.

Traverse the valid vehicle IDs in the workIDs list, calculate the positional difference (i.e., pixel distance) between the current frame and the previous frame of the vehicle, and convert it into the true spatial distance. At the same time, based on the pixel width of the vehicle category, video frame rate, and the mapping relationship between pixel distance and actual spatial distance, the current frame speed of the vehicle is calculated, and the actual speed is obtained by combining the speed of the previous few frames for smoothing.

Store the vehicle speed and its corresponding ID in km/h with one decimal place in the speed two-dimensional list, and mark the vehicle ID and speed information in each frame of the image, as shown in Figure 10.

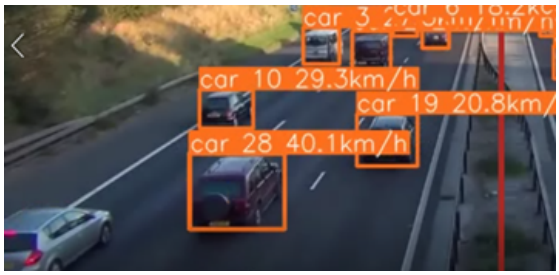


Figure 10. Mark the speed detection diagram

To verify whether the vehicle is speeding in practical applications, the vehicle ID and speed information are stored in a CSV file for subsequent queries and analysis, as shown in Table 6.

Table 6. CSV partial data

ID	Speed
10	49
11	52.8
12	14.7
15	27.4
19	50.6
20	15.1
21	22.7
22	61.4

Finally, save the marked speed information video and CSV file for subsequent vehicle speed analysis and statistics.

(a) Daytime speed measurement diagram

(b) Cloudy speed measurement diagram

(c) Night speed measurement diagram

(d) Rainy night speed measurement diagram

Figure 11. Speed measurement diagram

(a) Before camera movement (b) After camera movement

Figure 12. Error Examples

4.4 Comparative Experiment

This paper demonstrates the speed measurement performance of the algorithm under different lighting, weather, and road conditions by selecting four different scenes: daytime, cloudy, rainy night, and nighttime for video detection. As shown in photos of Figure 11, through result analysis, we can find that in these four scenarios, the algorithm can stably and accurately measure vehicle speed. This indicates that the algorithm has excellent accuracy and robustness, and can demonstrate excellent speed measurement performance in different scenarios. This algorithm can maintain stable speed measurement, whether at night with dim lighting or in complex weather conditions such as rainy days. This algorithm has a very broad application prospect in fields such as traffic management, urban planning, and public safety.

Although in certain scenarios, such as the placement of video cameras, deviation of vehicle movement direction, and vehicle pauses, this method may have certain errors, as shown in Figure 12. The vehicle with ID 17 in the picture did not move, but due to the movement of the camera, the position of the vehicle changed, resulting in a speed increase for vehicle 17. But in general, this method can be used as a simple and effective way to estimate vehicle speed.

5 Conclusion

This paper provides an overview of vehicle detection, tracking, and speed measurement work, introducing relevant technologies and algorithms in this field, and using the YOLOv5 algorithm + DeepSORT algorithm for vehicle detection, tracking, and speed measurement. I have mainly completed the following three parts of work:

(1) In the vehicle detection section, this paper first analyzes the YOLO series of algorithms and focuses on the YOLOv5 algorithm and its different variants. The YOLOv5 algorithm variant from the COCO dataset was used for training, and the YOLOv5s model was ultimately selected for subsequent experiments.

(2) In the vehicle tracking section, this paper provides a detailed analysis of the principles of the DeepSORT algorithm, including the Hungarian algorithm and the Kalman wave filtering algorithm used, and describes the specific process. Make changes to the deep grid structure and train accordingly. Finally, multiple videos were used for testing.

(3) In the section about vehicle speed measurement, this paper mainly designs an algorithm to convert the pixel distance between two frames into the actual distance for speed measurement.

The future research directions of this article are as follows:

(1) Enhance the dataset by testing the accuracy of the model with a dataset that includes more vehicles and scenarios.

(2) Improve the model. In order to improve the accuracy of prediction results, more suitable models can be designed by improving the model algorithm. In neural network models, the number of neural nodes and network layers has a significant impact on the prediction results. Therefore, when the computer configuration is high, the number of neural nodes and network layers can be appropriately increased, and the input data volume and configuration parameters can be adjusted to obtain more accurate prediction results.

(3) Apply the algorithm to other aspects. It can detect license plates, replace ID values in speed measurement, and directly obtain vehicle information, which is more convenient; repositioning the incoming video with surveillance allows for real-time monitoring; perform traffic light detection to adjust the time ratio of traffic lights and alleviate traffic congestion.

(4) Make it a system. Transform all functions into an intelligent urban traffic management system that can be used by relevant departments.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Confilict of Interest

The authors declare that they have no conflicts of interest.

References

[1] National Bureau of Statistics, "Statistical communiqué of the People's Republic of China on the 2022 national economic and social development," 2023. https://www.stats.gov.cn/sj/zxfb/202302/t20230228_1919011.html

[2] Y. Wang and H. Zhong, "Mitigation strategies for controlling urban particulate pollution from traffic congestion: Road expansion and road public transport," *J. Environ. Manage.*, vol. 345, p. 118795, 2023. https://doi.org/10.1016/j.jenvman.2023.118795

[3] Z. Zheng, Z. Wang, L. Zhu, and H. Jiang, "Determinants of the congestion caused by a traffic accident in urban road networks," *Accid. Anal. Prev.*, vol. 136, p. 105327, 2020. https://doi.org/10.1016/j.aap.2019.105327

[4] W. H. Lee and C. Y. Chiu, "Design and implementation of a smart traffic signal control system for smart city applications," *Sensors*, vol. 20, no. 2, p. 508, 2020. https://doi.org/10.3390/s20020508

[5] A. Essien, I. Petrounias, P. Sampaio, and S. Sampaio, "A deep-learning model for urban traffic flow prediction with traffic events mined from twitter," *World Wide Web*, vol. 24, no. 4, pp. 1345–1368, 2021. https://doi.org/10.1007/s11280-020-00800-3

[6] C. Y. Cao, J. C. Zheng, Y. Q. Huang, J. Liu, and C. F. Yang, "Investigation of a promoted you only look once algorithm and its application in traffic flow monitoring," *Appl. Sci.*, vol. 9, no. 17, p. 3619, 2019. https://doi.org/10.3390/app9173619

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788. https://doi.org/10.1109/CVPR.2016.91

[8] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 6517–6525. https://doi.org/10.1109/CVPR.2017.690

[9] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018. https://doi.org/10.48550/arXiv.1804.02767

[10] Y. Yin, H. Li, and W. Fu, "Faster-YOLO: An accurate and faster object detection method," *Digit. Signal Process.*, vol. 102, p. 102756, 2020. https://doi.org/10.1016/j.dsp.2020.102756

[11] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020. https://doi.org/10.48550/arXiv.2004.10934

[12] G. Yang, W. Feng, J. Jin, Q. Lei, X. Li, G. Gui, and W. Wang, "Face mask recognition system with YOLOV5 based on image recognition," in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2020, pp. 1398–1404. https://doi.org/10.1109/ICCC51575.2020.9345042

[13] C. Chen, Y. Fan, and L. Wang, "Logo detection based on improved mosaic data enhancement and feature fusion," *Comput. Measur. Control*, vol. 30, no. 10, pp. 188–194, 2022.

[14] D. Walawalkar, Z. Shen, Z. Liu, and M. Savvides, "Attentive CutMix: An enhanced data augmentation approach for deep learning based image classification," *arXiv preprint arXiv:2003.13048*, 2020. https://doi.org/10.48550/arXiv.2003.13048

[15] M. Gao, Y. Du, Y. Yang, and J. Zhang, "Adaptive anchor box mechanism to improve the accuracy in the object detection system," *Multimed. Tools Appl.*, vol. 78, pp. 27 383–27 402, 2019. https://doi.org/10.1007/s11042-019-07858-w

[16] Y. Chen and B. Wu, "Multi-target tracking algorithm based on YOLO+ DeepSORT," *J. Phys.: Conf. Ser.*, vol. 2414, no. 1, p. 012018, 2022. https://doi.org/10.1088/1742-6596/2414/1/012018

[17] Y. Pei, H. Liu, and Q. Bei, "Collision-line counting method using DeepSORT to count pedestrian flow density and hungary algorithm," in *2021 IEEE 3rd International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, Changsha, China, 2021, pp. 621–626. https://doi.org/10.1109/ICCASIT53235.2021.9633356

[18] S. Feng, K. Hu, E. Fan, L. Zhao, and C. Wu, "Kalman filter for spatial-temporal regularized correlation filters," *IEEE Trans. Image Process.*, vol. 30, pp. 3263–3278, 2021. https://doi.org/10.1109/TIP.2021.3060164

[19] Y. Ma, Z. Zhang, and A. Ihler, "Multi-lane short-term traffic forecasting with convolutional LSTM network," *IEEE Access*, vol. 8, pp. 34 629–34 643, 2020. https://doi.org/10.1109/ACCESS.2020.2974575

[20] W. Zhong, Y. Q. Jiang, and X. Zhang, “Research on road object detection algorithm based on YOLOv5+ deepsort,” in *2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*, Xi’an, China, 2022, pp. 644–648. https://doi.org/10.1109/ICICML57342.2022.10009649