Research on road vehicle and lane line detection based on improved YOLOv4-tiny algorithm

Hanwen Liu[1], *Hongxia* Wang[1,*], *Kui* Zhou[2] and *Yun* Long[1]

[1]School of Mechanical Engineering, Hubei Institute of Automotive Technology, Shiyan, China
[2]School of Automotive Engineers, Hubei Institute of Automotive Technology, Shiyan, China

Abstract. In order to improve the speed of road vehicle and lane line detection, a road vehicle and lane line detection method with improved YOLOv4-tiny algorithm is proposed by using K-means++ clustering algorithm instead of the original K-means algorithm. Using the Mosaic data enhancement method, four images are randomly deflated and then stitched into one image to enrich the detection target background. The optimal weight values are derived by multi-scale training using the GPU through image feature extraction of the BDD10K dataset to achieve the detection of vehicle and lane line targets in the images. The results show that the improved YOLOv4-tiny algorithm achieves a detection speed of 134 FPS and an average accuracy of 77.84% mAP in highway lane line detection. After comparison, the detection speed of the improved algorithm is significantly improved, effectively improving the efficiency of highway vehicle and lane line detection.

Keywords: Clustering algorithm, Improved YOLOv4-tiny algorithm, Lane line detection, Deep learning.

1 Introduction

With the rapid development in the field of artificial intelligence, a large number of intelligent assistive devices are used in vehicles. Lane line detection, as one of the key technologies for autonomous driving and intelligent assisted driving, is widely used in driverless systems, vehicle trajectory deviation warning systems and anti-collision systems. Due to the specificity of the use scenario, improving the detection accuracy and detection speed has important research value for road safety.

In recent years, with the exploration of deep learning techniques, target detection algorithms have shifted from traditional machine vision-based[1] methods to deep learning[2-4] neural network-based methods. Deep learning algorithm-based lane line detection performs feature extraction, image classification, and target detection on images through a convolutional neural network [5] (CNN). A standard convolutional neural network abstracts the raw information into a feature representation for the target task through a series of passes such as convolutional layers, pooling layers and non-linear activation function

[*] Corresponding author: 8784145@163.com

mapping. In [6], the computational speed is enhanced by performing top-down and hierarchical attention distillation within the network itself to enhance the learning of representations in the CNN-based lane detection model. In [7], a feature map information fusion method based on VGG networks is proposed, introducing the concept of null convolution to identify lane lines in complex environments. In [8], the adaptive double-threshold screening method is proposed to extract lane line features in order to fit lane lines. In [9], the proposed improved YOLOv3[10] algorithm, using the K-means++ clustering algorithm, determines the number of target a priori frames and the corresponding width and height values according to the inherent characteristics of highway lane line width and height, so as to detect the lane lines, shortening the detection time and effectively avoiding the problem of missed detection. In [11], the multi-label detection approach is proposed to establish constraint relationships between multiple labels to obtain more complete vehicle information and add an input layer to YOLOv4 [12] to improve the network detection speed.

In complex traffic environments, the speed of target detection is particularly important, The deep network structure of YOLOv4 allows the algorithm to have a high detection accuracy, but it also needs to sacrifice a lot of time for processing a large number of parameters; YOLOv4-tiny retains the YOLOv4 backbone network structure and simplifies the feature extraction layer, which significantly improves the computational speed due to the greatly reduced number of parameters, but decreases the detection accuracy. In order to ensure the detection accuracy and improve the detection speed, this paper will improve the lightweight YOLOv4-tiny network structure, aiming to improve the computation speed and achieve the detection of road vehicles and lane lines.

2 Analysis of the YOLOv4-tiny algorithm

2.1 YOLOv4-tiny network architecture analysis

The YOLOv4-tiny network structure is shown in Figure 1. One of the backbone feature extraction layers (CSPDarknet53-tiny) first performs CBL processing on the input image. The CBL structure is shown in Figure 1(a) and includes Conv (convolutional layer), BN (batch normalisation), and LeakyRelu (activation function). Two CBL layers compress the image in height and width, and then the image is further compressed and downsampled by a network structure with three sets of residual and pooling layers (Maxpool) stacked on top of each other. The residual layer incorporates the features of the CSP (Cross Stage Partial Connections) structure, as shown in Figure 1(b), where the CSP structure adds another residual edge next to the main line channel made up of the residual components, and the two channels are combined and output. The enhanced feature extraction layer adopts a feature pyramid network (FPN) structure, outputting two effective feature layers with higher semantic information for the final detection of image targets, where a represents the number of prediction frames and c represents the prediction category, and the two channels are used to detect smaller and larger targets respectively.

2.2 Loss function calculation

The YOLOv4-tiny algorithm takes into account the penalty term for the aspect ratio of the prediction frame and proposes CIoU with the following equation.

$$CIoU = IoU - \frac{\rho^2(b, b^{gt})}{c^2} - \alpha v \tag{1}$$

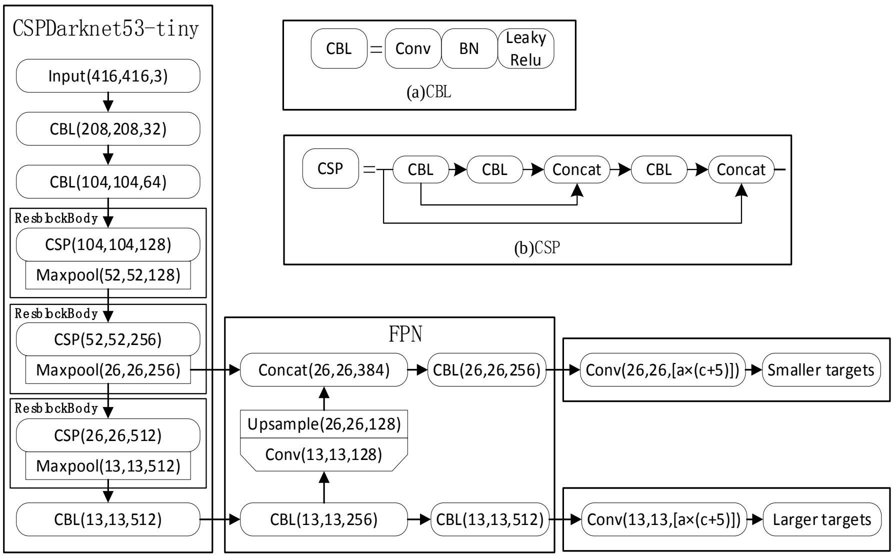


Fig. 1. YOLOv4-tiny network architecture.

where $\rho^2(b,b^{gt})$ is the Euclidean distance between the centre point of the prediction frame and the real frame, and c is the diagonal distance between the prediction frame and the smallest closed area of the real frame, as shown in Figure 2. α,ν is the influence factor, α the parameter is used to balance the ratio, and the ν parameter is used to measure the consistency of the aspect ratio with the following equation.

$$\alpha = \frac{\nu}{1-IoU+\nu} \tag{2}$$

$$\nu = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2 \tag{3}$$

CIoU takes into account the overlap rate, distance and scale between the prediction frame and the target, making the final target frame regression more stable and the loss function more inclined to optimize in the direction of more overlapping regions.

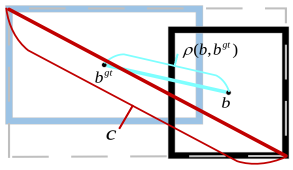


Fig. 2. Schematic diagram of CIoU parameters.

From the above equation, the confidence level can be calculated as:

$$confidence = \Pr(object) \times CIoU \tag{4}$$

Assuming the image is divided into n×n grid cells, the position of the prediction box and the width and height parameters are adjusted using the distance between the centre of the a priori box and the real box, $\Pr(object)$ the adjustment process $confidence$ is as follows.

$$\begin{cases} x = sigmoid(x_{data}) + x_{grid} \\ y = sigmoid(y_{data}) + y_{grid} \\ w = e^{w_{data}} \times w_{anchor} \\ h = e^{h_{data}} \times h_{anchor} \end{cases} \quad (5)$$

where x_{data}, y_{data} is the distance between the a priori box and the centre point of the real box, x_{grid}, y_{grid} is the co-ordinate of the centre point of the a priori box, w_{data}, h_{data} is the adjustment parameter of the width and height of the a priori box, w_{anchor}, h_{anchor} is the width and height of the a priori box, and x, y, w, h is the adjusted position and width and height parameter of the prediction box. The final prediction box is obtained by pre-setting the confidence threshold, removing the a priori boxes with confidence below the threshold, and then non-maximum suppression for the a priori boxes with confidence above the threshold.

The loss function expresses the difference between the predicted value and the true value by comparing the predicted value output after the image has passed through the network with the true value in the image, thus serving as an important parameter to measure the effectiveness of network detection. From equation (1), the loss function can be calculated as follows.

$$loss = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \quad (6)$$

3 YOLOv4-tiny algorithm improvements

The improved YOLOv4-tiny algorithm uses the K-means++ clustering algorithm to replace the original clustering algorithm, reducing the time consuming network model operations while increasing the accuracy rate. The figure3 shows the improved YOLOv4-tiny algorithm target detection process, and this section will analyse the clustering algorithm and the K-means++ optimisation theory in detail respectively.

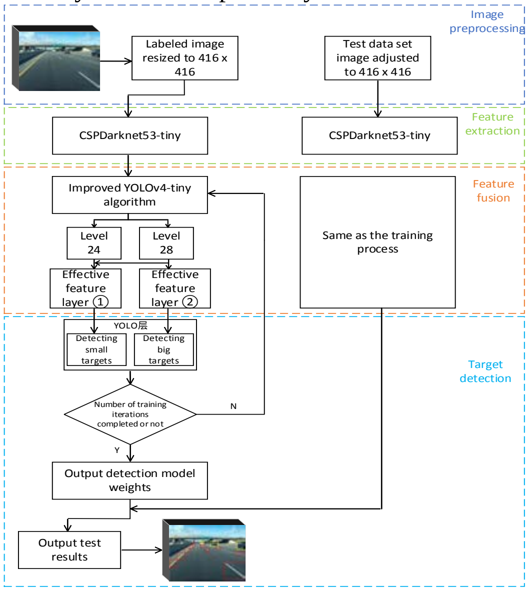


Image preprocessing
Labeled image resized to 416 x 416
Test data set image adjusted to 416 x 416
Feature extraction
CSPDarknet53-tiny
CSPDarknet53-tiny
Feature fusion
Improved YOLOv4-tiny algorithm
Same as the training process
Level 24
Level 28
Effective feature layer ①
Effective feature layer ②
YOLO层
Detecting small targets
Detecting big targets
Target detection
Number of training iterations completed or not
N
Y
Output detection model weights
Output test results

Fig. 3. Flow chart of target3 detection.

3.1 Clustering centre generation

The YOLOv4-tiny algorithm uses a K-means clustering algorithm based on Euclidean distance. The core idea is to randomly select K points as the initial clustering center, calculate the distance from each of the remaining points to the clustering center, and classify the object into the class in which the nearest clustering center is located. Since the algorithm relies on the selection of the initial K-value, the K-value has a very strong influence on the final clustering result and the computation time. Once the initial value is poorly chosen, it is easy to fall into the local optimal solution, which affects the detection accuracy. On the other hand, the K-means algorithm requires constant sample classification adjustment, and the time overhead can be very large when the data volume is large.

3.2 Improved K-means++ algorithm

In order to improve the detection performance and optimise the training effect, this paper uses the K-means++ clustering algorithm to cluster the vehicle and lane line features. The greater the distance, the higher the probability of being selected as the next cluster centre, until K cluster centres have been selected. By using the less random K-means++ clustering algorithm, we can effectively avoid the situation of falling into a local optimum solution due to the initial clustering centre error, and also solve the problem of increasing time overhead caused by repeatedly adjusting the sample classification.

3.3 A priori frame design

The YOLOv4-tiny algorithm uses anchor boxes in the prediction process, which is a clustering algorithm that generates K square boxes with fixed width and height values based on the position and width of the bounding boxes with the detected target in the image. By comparing the differences between the anchor boxes and the a priori boxes, the width and height parameters of the a priori boxes are continuously updated to obtain accurate prediction boxes containing the target.

In the process of updating the parameters of the prior frame using the clustering algorithm, the traditional clustering method is to use the Euclidean distance to measure the error, but when the size of the prior frame is larger, the error value will then become larger, so the intersection ratio (IoU) of the picture sample label frame and the prior frame is used as the objective function to express the difference between the clustering centre and the sample, the larger the intersection ratio, the smaller the distance, the better the clustering effect. The distance formula is.

$$D = \min \sum_{box=0}^{n} \sum_{cen=0}^{c} [1 - IoU] \tag{7}$$

where box is the picture sample label box, cen is the cluster centroid,n is the number of picture samples and c is the number of categories.

The BDD100K dataset images used in this paper are scaled down to a size of 416×416, and the corresponding target selection boxes in the figure are adjusted for width and height, and the results are shown in the figure4 after K-means++ clustering of the dataset samples.

From the figure, it can be seen that the K-value tends to be smooth at 6, with an accuracy rate of about 71%. Therefore, in this paper, six a priori frames are selected to identify the targets, and the clustering centroids are [10,12], [16,23], [25,13], [28,34], [48,58] and [102,126].

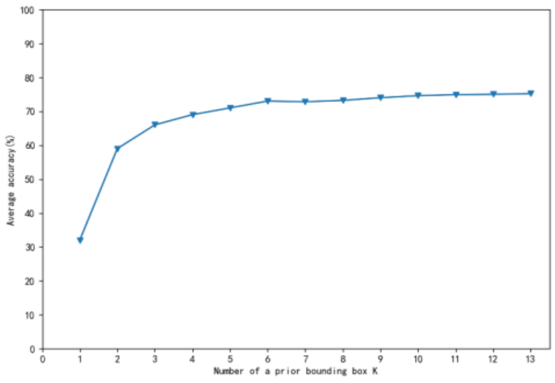


Fig. 4. Plot of number of a 5priori frames against average accuracy.

3.4. A priori frame assignment

The YOLOv4-tiny algorithm uses a multi-scale fusion method for target detection, which enhances the accuracy of detecting small targets and is suitable for distant target recognition in road scenes. the YOLOv4-tiny algorithm fuses two scales, 1×3 13and 2×626, and assigns three a priori frames to each scale. Based on the clustering results in the previous section, six clustering centres are assigned to the two scales in this paper, with the smaller the scale, the higher the accuracy of detecting large targets, as shown in Table 1.

Table 1. Table of a priori frame assignments for different scales.

Scale	A priori box
13×13	[28,34], [48,58], [102,126]
26×26	[10,12], [16,23], [25,13]

The improved YOLOv4-tiny algorithm reduces the amount of operations and improves the real-time detection while ensuring detection accuracy, enabling road scene detection.

4 Testing and analysis of results

4.1 Data set production

In this experiment, the BDD100K open driving dataset is used. Since only vehicles and lane lines are used as detection targets in this experiment, the training sample images are reduced to 6000 images, of which 80% are used for training and 20% are used for validation. The vehicles and lane lines in the training set images are labeled using the LabelImg image annotation tool to generate the corresponding label files.

4.2 Test environment

The experimental environment in this paper is Windows 10 64-bit operating system, using AMD Ryzen 7 5800H with Radeon Graphics 3.20GHz processor, computer graphics configuration is NVIDIA GeForce RTX3060 Laptop GPU, 6G memory. To make full use of the GPU and accelerate the training process, CUDA11.1 was installed on the system, as well as the corresponding cudnn. the deep learning framework was pytorch v1.8.0. The initial weight parameters used coco dataset weights to accelerate the loss function descent, and the number of training iterations was 800times. the trend of the loss function during the training process was recorded as shown in Figure.5

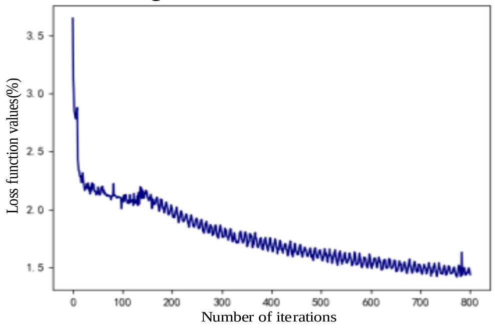


Fig. 5. Loss function change curve.

4.3. Test results and analysis

As can be seen from the figure 5, the loss function loss value is about 3.64% at the beginning of the training, with the increase of the number of training iterations, the loss function value gradually decreases, and the loss function value is about 1.36% when iterating to800 the second, reaching the vehicle and lane line detection requirements. Figure 6 shows the detection effect of the improved YOLOv4-tiny algorithm for real road scenes, including four driving environments: daytime, darkness with street lights, darkness without street lights, and tunnel, respectively. The results show that the improved YOLOv4-tiny algorithm has good detection effect, and the algorithm has good adaptability and can accurately identify vehicles and lane lines in complex real-time environments with poor lighting and vision conditions.

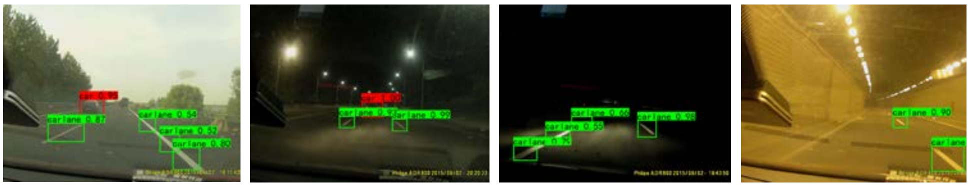


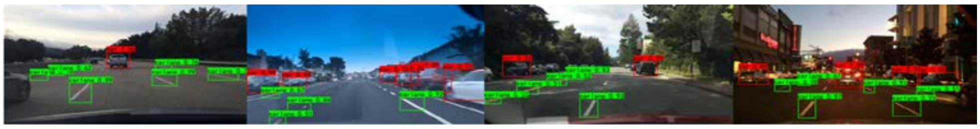
Fig. 6. Detection results of the improved7 S-YOLOv4-tiny algorithm.

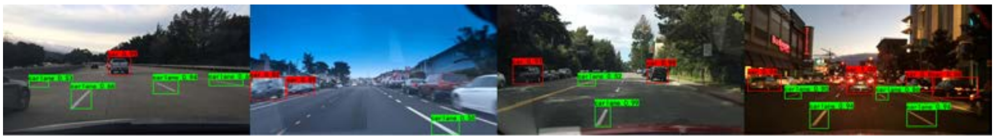
The performance of different algorithms is evaluated by comparing the mAP and FPS of different network models. mAP value is the average accuracy of all detection categories, higher mAP means better detection accuracy; FPS value is the number of frames per second transmitted from the image, larger FPS value means faster image detection and smoother picture. The improved algorithm (S-YOLOv4-tiny), while retaining the speed of YOLOv4-tiny lightweight network computing, optimises the original clustering algorithm and improves the detection accuracy of the algorithm. A comparison with the existing algorithm verifies the effectiveness of the algorithm improvements, and a comparison of the network structure performance is shown in Table 2.

The data in the table shows that the lightweight S-YOLOv4-tiny algorithm has a faster detection speed, and the FPS values have improved to different degrees compared to YOLOv4 and YOLOv4-tiny algorithms; compared to before the improvement, the mAP value of S-YOLOv4-tiny algorithm has improved by 14.73%, which is only 3.69% lower than that of YOLOv4 algorithm. The figure7 shows the comparison of the detection results of different network models. It can be seen that S-YOLOv4-tiny has significantly improved the detection effect compared with the other two algorithms, and is more effective in detecting complex environments

Table 2. Network architecture performance comparison.

Network Model	YOLOv4	YOLOv4-tiny	S-YOLOv4-tiny
Number of participants	6.4 × 107	5.91 × 106	5.91 × 106
Calculated volume	2.99 × 1010	3.42 × 109	3.42 × 109
mAP (%)	81.53	63.11	77.84
FPS	54	129	134


(a) S-YOLOv4-tiny test results


(b) YOLOv4-tiny test results

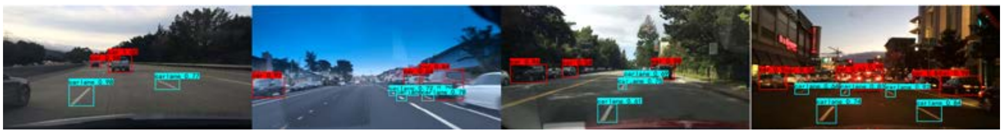

(c) YOLOv4 test results

Fig. 7. Comparison of test results.

5 Conclusion

The K-means++ clustering algorithm was used to optimise the original network to achieve the detection of vehicles and lane lines in highway scenes. The improved algorithm achieved a detection speed of 134 FPS and an average accuracy of mAP of 77.84% in highway lane line detection, which effectively improved the accuracy and speed of highway vehicle and lane line detection in comparison with the existing algorithm and verified the effectiveness of the algorithm.

References

1. T H Yu, R B Wang, B Y Gu, L Guo. J Highw Transp Res Dev.**01**,139-142+158,(2006)
2. D C Wang, C H Bai, K J Wu. J Front Comput Sci Technol.**1**-15 (2021)
3. X Wu, X R Song, S Gao, C B Chen. Transd Micros Technol.**40**(02),4-7+18,(2021)
4. D G Xu, L Wang, F Li. Comput Eng Appl,**57**(08),10-25(2021)
5. A Krizhevsky, I Sutskever, G Hinton. NIPS,**25**(2),(2012)
6. Y Hou，Z Ma，C Liu，CC Loy,(2019)
7. Y Gao, C Wang, Z J Li. S T A E,**21**(24),10401-10406,(2021)

8. Y Zhao, J G Zhao. S T A E,**21**(07),2782-2787,(2021)
9. W L Cui, Y J Wang, S Q Kang, J B Xie, Q Y Wang, V.I. Mikulovich. JAS,**1-9,(**2021)
10. J Redmon, A Farhadi. arXiv e-prints,(2018)
11. Y X Wang, H S Song, H X Liang, L Y Yu, Y Xu. Comput Eng Appl,**57**(13),218-226,(2021)
12. A Bochkovskiy, C Y Wang, H Liao.CVPR,(2020)