Light-weighted vehicle detection network based on improved YOLOv3-tiny

Pingshu Ge¹, Lie Guo²,³, Danni He² and Liang Huang²

Abstract
Vehicle detection is one of the most challenging research works on environment perception for intelligent vehicle. The commonly used object detection network is too large and can only be realized in real-time on a high-performance server. Based on YOLOv3-tiny, the feature extraction was realized using light-weighted networks such as DarkNet-19 and ResNet-18 to improve accuracy. The K-means algorithm was used to cluster nine anchor boxes to achieve multi-scale prediction, especially for small targets. For automotive applicable scenarios, the proposed vehicle detection network was executed in an embedded device. The KITTI data sets were trained and tested. Experimental results show that the average accuracy is improved by 14.09% compared with the traditional YOLOv3-tiny, reaching 93.66%, and can reach 13 fps on the embedded device.

Keywords
Intelligent vehicle, vehicle detection, light-weighted network, YOLOv3-tiny, residual network

Date received: 14 September 2021; accepted: 10 January 2022

Handling Editor: Peio Lopez Iturri

Introduction
With the growth of the economy, the automotive ownership of China has continued to increase, reaching 281 million by the end of 2020.¹ When bringing convenience to human life, automobiles have also generated problems, such as environmental pollution, traffic congestion, and travel safety.² In order to effectively protect the safety of drivers, passengers, and pedestrians, domestic and foreign manufacturers have started to conduct research works on automotive safety driving-assisted technology and intelligent vehicle. Those intelligent systems can take over control of vehicles when necessary and reduce traffic accidents. Among them, environment perception technology is a prerequisite for achieving vehicle intelligence.³ As an essential component of environment perception for intelligent vehicle, vehicle detection has also attracted the attention from scholars at home and abroad.⁴

Vehicle detection based on vision sensors can be divided into three types, such as feature-based, machine learning-based, and deep learning-based. Feature-based vehicle detection method refers to using fixed features of the vehicle, such as symmetry, lights, and shadows, to identify the vehicle.⁵ This method is characterized by fast computing speed, but it also is greatly influenced by the environment and shows poor algorithm robustness.⁶

¹College of Mechanical & Electronic Engineering, Dalian Minzu University, Dalian, China
²School of Automotive Engineering, Dalian University of Technology, Dalian, China
³Ningbo Institute of Dalian University of Technology, Ningbo, China

Corresponding author:
Lie Guo, School of Automotive Engineering, Dalian University of Technology, No. 2 Linggong Road, Ganjingzi District, Dalian 116024, Liaoning, China.
Email: guo_lie@dlut.edu.cn

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
Machine learning-based vehicle detection method refers to the extraction of features from many samples and the use of classification algorithms to train the classifier. However, the time complexity of the sliding window algorithm used in its prediction is high, computationally intensive and the detection frame regression is not accurate enough. Deep learning is currently the most widely used method in the machine vision field because of its fast computing speed, high accuracy, and the ability to identify multiple species at the same time.8 With the development of deep learning and GPU, target detection technology based on deep learning is more and more widely used.9,10 However, several commonly used detection network data are too large and can only be realized in real-time on a high-performance server. Therefore, implementing vehicle detection in embedded devices is also one of the challenges of current research.11

With the development of artificial intelligence and deep learning, the convolutional neural network has become the mainstream method in object detection. The object detection based on convolutional neural network has also obtained good results on some public data sets. Currently, there are two main types of convolutional neural networks for object detection: a two-stage detection algorithm represented by region-based convolutional neural network (R-CNN),12 and an one-stage detection algorithm represented by you only look once (YOLO)13 and single shot multi-box detector (SSD).14 Faster R-CNN, as a representative of two-stage algorithms, can extract features from the original image using visual geometry group (VGG) networks and connect region proposal networks (RPNs) to generate several detection regions. In this way, it can classify the object in these detection regions to obtain the results.15 YOLO, as a representative of the one-stage algorithms, generates a $7 \times 7 \times 30$ tensor by convolving the original image into a $7 \times 7$ grid. Each grid predicts the probability of two bounding boxes and the class to which they belong. The results are finally obtained using a non-maximum suppression algorithm. YOLOv3 combines YOLO algorithm for dividing the grid prediction and SSD for multi-scale prediction, using a DarkNet-53 network to abstract features, and producing prediction results at three different scales.16 From the performance of these four algorithms on the VOC data set, the detection speed of the two-stage model Faster R-CNN is only 7 fps, which cannot meet the real-time performance. The detection accuracy of the one-stage model YOLOv3 is the highest. Therefore, there are many networks are based on Yolov3 to improve their performance.17,18 Although all those one-stage models can operate at high fps on servers, it is difficult to achieve real-time detection in embedded devices.19

The main purpose is to realize vehicle detection running on embedded devices. Therefore, the YOLOv3-tiny model is used as the base network. YOLOv3-tiny is proposed to improve the detection accuracy of the model by optimizing the feedforward network. Because the model is ineffective in detecting small objects, nine anchor boxes are clustered using the $K$-means algorithm and multi-scale experiments are conducted. Finally, the detection performance of the model is tested on an embedded device TX2 development board and a laptop with an NVIDIA GTX1060 graphics card. The key contributions of our work are as follows: (1) nine anchor boxes are clustered using the $K$-means algorithm and the multi-scale experiments are conducted and (2) the vehicle detection algorithm is transplanted to an embedded device TX2 and can achieve real-time detection speed.

**Vehicle detection light-weighted network**

**Vehicle detection network YOLOv3-tiny**

The designed vehicle detection network is applicable to advanced driver assistance driving systems. The traditional YOLOv3 network is designed for multi-object detection, which results in poor real-time performance on embedded devices. YOLOv3-tiny is an improved version of YOLOv3 network, whose network structure is small but inherits the advantages of traditional YOLOv3 algorithm completely.20 It brings fast detection speed with a small network. The network structure of YOLOv3-tiny is shown in Figure 1.

The feature extractor of YOLOv3-tiny network consists of seven convolutional layers and six maximum pooling layers, which replaces the backbone network of YOLOv3.21 The kernel size of each convolutional layer is $3 \times 3$, and the stride of each pooling layer is 2. The size of the input data is $416 \times 416 \times 3$, and the size of the output tensor obtained after the feature extractor is $13 \times 13 \times 256$. A feature map of $26 \times 26 \times 128$ is obtained using ascending sampling in this layer. A combination of the bottom feature map and the upper feature map can achieve high-level semantic information. The output size after feature combination is $26 \times 26 \times 384$. By this means, the YOLOv3-tiny network can realize object prediction in two scales; they are $13 \times 13$ and $26 \times 26$. The smaller scale with $13 \times 13$ can detect larger objects, which means that the feature map is downsampled by 32 times. On the contrary, the larger scale with $26 \times 26$ can detect smaller objects, which means that the feature map is downsampled by 16 times.

The final output vector size of the network is determined by the type of the predicted objects, that is, $B \times B \times (n_{anchor} \times (5 + n_{type}))$. $B$ represents the number of grids. $n_{anchor}$ represents the number of prior boxes to be predicted for each grid. The number 5 represents the size for five parameters, which are the
The confidence degree $c$ of the predicted bounding box, the offset of the central position of the bounding box relative to the grid in horizontal and vertical direction, respectively, and the ratio of the bounding box relative to the width and height of the original image, respectively. $n_{\text{type}}$ represents the number of classes to be predicted. In this article, the object types to be predicted include “car,” “bus,” and “truck,” thus $n_{\text{type}} = 3$. The number of prior boxes to be predicted in each grid is 3, thus $n_{\text{anchor}} = 3$. The final output vector size of the network at a scale of 13 $\times$ 13 is 13 $\times$ 13 $\times$ (3 $\times$ (5 + 3)), that is, 13 $\times$ 13 $\times$ 24. The final output vector size of the network at a scale of 26 $\times$ 26 is 26 $\times$ 26 $\times$ (3 $\times$ (5 + 3)), that is, 26 $\times$ 26 $\times$ 24. After obtaining the prediction results on these two scales, the non-maximum suppression (NMS) algorithm is applied to get the final results. \(^{22}\)

The loss function mainly includes three parts: position error, confidence error, and classification error, \(^{23}\) which are defined as follows

$$E_{\text{loss}} = E_{\text{coord}} + E_{\text{iou}} + E_{\text{cls}} \quad (1)$$

where $E_{\text{loss}}$ is the total error, $E_{\text{coord}}$ is the position error, $E_{\text{iou}}$ is the confidence error, and $E_{\text{cls}}$ is the classification error.

The position error is calculated using the width, height, and center position of the object with the following formula

$$E_{\text{coord}} = \lambda_{\text{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{D} I_{ij}^\text{obj} \left[ (\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2 \right] + \lambda_{\text{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{D} I_{ij}^\text{obj} \left[ \sqrt{w_i} - \sqrt{\hat{w}_i} \right]^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \quad (2)$$

where $\lambda_{\text{coord}}$ is the weight coefficient of position error, $I_{ij}^\text{obj}$ represents the $j$th candidate box in the $i$th grid is responsible for the object, $(x_i, y_i)$ is the true center position of the object, $(\hat{x}_i, \hat{y}_i)$ is the center position of the predicted object, $s^2$ is the number of grids in the output layer of the network, $D$ is the number of candidate boxes in each grid, $w_i$ and $\hat{w}_i$ are the width of the true and predicted object, respectively, and $h_i$ and $\hat{h}_i$ are the height of the true and predicted object, respectively. Since the size difference of each object may be large, the square root of width and height is used to calculate the position error to balance the ratio of each error.

Considering that many grids do not have objects in them, different weight coefficients are set to balance the confidence for those bounding boxes where existing
objects and without existing objects. Here, cross-entropy is utilized to calculate the confidence error

\[
E_{\text{iou}} = -\alpha_{\text{coord}} \sum_{i=0}^{c} \sum_{j=0}^{D} l_{ij}^{\text{obj}} \left[c_{ij} \log(\hat{c}_{ij}) + (1 - c_{ij}) \log(1 - \hat{c}_{ij})\right] \\
- \lambda_{\text{noobj}} \sum_{i=0}^{c} \sum_{j=0}^{D} l_{ij}^{\text{noobj}} \left[c_{ij} \log(\hat{c}_{ij}) + (1 - c_{ij}) \log(1 - \hat{c}_{ij})\right]
\]

(3)

where \(l_{ij}^{\text{obj}}\) denotes the \(j\)th candidate box in the \(i\)th grid does not exist object, \(\lambda_{\text{noobj}}\) is the weight coefficient of confidence error, and \(c_{ij}\) and \(\hat{c}_{ij}\) are the true and predicted confidence value for the \(j\)th bounding box in the \(i\)th grid.

Classification error is used to describe the classification error of an object, which exists only when the \(j\)th candidate box of the \(i\)th grid is responsible for a specific target, that is

\[
E_{\text{cls}} = -\sum_{i=0}^{c} \sum_{j=0}^{D} l_{ij}^{\text{obj}} \sum_{c \in \text{classes}} \left[P_{ij}(c) \log(\hat{P}_{ij}(c)) + (1 - P_{ij}(c)) \log(1 - \hat{P}_{ij}(c))\right]
\]

(4)

where \(P_{ij}(c)\) and \(\hat{P}_{ij}(c)\) are the true and predicted probability of class \(c\), respectively.

A large amount of data is required for the training of the detection network based on deep learning. The detection object of this article is mainly concentrated on vehicles running on the road. The training data are extracted from the public data set KITTI, the most popular data set for autonomous driving research. This data set contains images of three kinds of vehicle categories, named, “Car,” “Van,” and “Truck.” From this data set, 6798 images are selected as the training data and the remaining 1565 images are used as the test data to verify the performance of the vehicle detection model. The vehicle detection network is trained using a server with NVIDIA RTX 2080Ti and Intel i9-9900k. The size of the input image is \(416 \times 416\). Totally, 120,000 iterations are performed during the training. The batch size is 32, and the learning rate is 0.001. The total loss curve during the training is shown in Figure 2.

It can be seen from Figure 2 that the training error decreases rapidly at the beginning of training. After the error decreases to 4, the convergence speed becomes slower. Then, the training error stays below 2.0 after 10,000 iterations. From 10,000 iterations to 100,000 iterations, the error decreases from about 2.0 to about 1.0, and then the error stays around 1.0 with a small floating range. To avoid overfitting, the training is stopped when the network reaches 120,000 iterations.

The object position accuracy is an important index to test the performance of the vehicle detection network. The intersection union ratio (IOU) is utilized to the positioning accuracy of the network. IOU is simply a ratio of area of overlap between the predicted bounding box and the ground-truth bounding box and the area encompassed by both the predicted bounding box and the ground-truth bounding box. Figure 3 shows the IOU accuracy of the vehicle detection network based on YOLOv3-tiny with scales of \(13 \times 13\) and \(26 \times 26\).

Figure 3 shows that the IOU accuracy gradually approaches the ideal value 1 from the low level of about 0.2 at the beginning for both scales. The scale of \(13 \times 13\) has a small number of grids, so the error of bounding box regression will be relatively large, but the IOU accuracy will eventually stabilize at the level of about 0.9. The IOU accuracy of the scale of \(26 \times 26\) is finally maintained above 0.95. Figure 4 shows some detection results using the vehicle detection network trained using YOLOv3-tiny.

Experimental results show that the vehicle detection the tiny network can accurately locate the vehicle in most cases. When the road shadow blocks the vehicle or the vehicle is small in the image, there will still be missed detection. The traffic signs will be false detected as vehicles sometimes. The reason for this situation may be that the feature extractor of YOLOv3-tiny network cannot enough extract the features of the vehicle, resulting in false detection and missed detection. Furthermore, the predicted bounding box does not coincide with the true bounding box very well. In other words, the position of the vehicle is not located accurately.

**Optimization of vehicle detection network**

In order to improve the accuracy of the vehicle detection network, a better feature extractor is necessary.
This article replaces the feature extractors in the traditional YOLOv3-tiny network by DarkNet-19 and ResNet-18 light-weight networks, respectively. They can improve the feature extraction capability of the original network without increasing the time-consuming.

DarkNet-19 network consists of 19 convolutional layers and 5 maximum pooling layers, originally proposed as a foundation feature extraction network for YOLOv2.1 A 1×1 convolutional kernel is utilized to compress the feature representation between 3×3 convolutional layers, which helps to speed up the convergence and normalize the model. DarkNet-19 can achieve Top 5 (91.2%) accuracy on ImageNet data set. The parameters of DarkNet-19 network are shown in Table 1.

ResNet network was proposed by He et al.27 in 2015. ResNet protects the integrity of information by directly passing the input signal to the output. It only needs to learn the difference between the input and output, simplifying the learning objective and difficulty and reducing the possibility of gradient disappearance.28 Two ResNet units can be utilized to simplify the computation. Two 3×3 convolutional layers can be cascaded to form a ResNet unit. The other kind of ResNet unit is to cascade three convolutional layers in the sequence of 1×1, 3×3, and 1×1. This article adopts the ResNet-18 network as the feedforward network to optimize the performance of the vehicle detection network. ResNet network inherits the ability of the residual network to extract image features and meets the requirements of light-weighting. The structure of ResNet-18 network is shown in Figure 5.

After analyzing the false bounding boxes and missed vehicle, it is found that the width of most bounding boxes is distributed around 100 pixels and the height is around 50 pixels. Although replacing the feature extraction network of the original network can enhance...
the accuracy of vehicle detection, the detection performance on small vehicles is still unsatisfactory. Therefore, a scale of $52 \times 52$ is added to the original two scales ($13 \times 13$ and $26 \times 26$) to improve the detection effect of the vehicle detection network for small objects. Each scale uses three anchor boxes to generate the prediction results. It is necessary to cluster nine anchor boxes on the training data using a clustering algorithm. K-means clustering algorithm is used to generate the anchor boxes.

This article uses K-means to divide the real width and height of the bounding box into nine groups. The center of each group is taken as the parameter of the final nine anchor boxes. The detailed clustering process is as follows:

Step 1. Select clustering center points randomly as $\mu_1, \mu_2, \mu_3, \ldots, \mu_k \in \mathbb{R}^n$, where $k$ is the number of clustering centers.

Step 2. For each class $i$, the intersection ratio with each cluster mass is calculated, and the class of the cluster mass corresponding to the largest intersection ratio value is the class to which the sample belongs

$$c^{(i)} = \arg \min_j ||x^{(i)} - \mu_j||^2$$

where $c^{(i)}$ denotes the intersection ratio of class $i$, $x^{(i)}$ is the position of all samples in class $i$, and $\mu_j$ is the initial clustering center of mass.

Step 3. After traversing all the samples, for each class, the median in this class is chosen to update the new initial clustering center, which can be calculated as

$$\mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

### Table 1. Parameters of DarkNet-19.

| Type        | Filters | Size/stride | Output   |
|-------------|---------|-------------|----------|
| Convolutional | 32      | 3x3/2       | 224x224  |
| Maxpool     |         |             | 112x112  |
| Convolutional | 64      | 3x3/2       | 112x112  |
| Maxpool     |         |             | 56x56    |
| Convolutional | 128     | 3x3         | 56x56    |
| Convolutional | 64      | 1x1         | 56x56    |
| Convolutional | 128     | 3x3         | 56x56    |
| Maxpool     |         |             | 28x28    |
| Convolutional | 256     | 3x3         | 28x28    |
| Convolutional | 128     | 1x1         | 28x28    |
| Convolutional | 256     | 3x3         | 28x28    |
| Maxpool     |         |             | 14x14    |
| Convolutional | 512     | 3x3         | 14x14    |
| Convolutional | 256     | 1x1         | 14x14    |
| Convolutional | 512     | 3x3         | 14x14    |
| Convolutional | 256     | 1x1         | 14x14    |
| Convolutional | 512     | 3x3         | 14x14    |
| Maxpool     |         |             | 7x7      |
| Convolutional | 1024    | 3x3         | 7x7      |
| Convolutional | 512     | 1x1         | 7x7      |
| Convolutional | 1024    | 3x3         | 7x7      |
| Convolutional | 512     | 1x1         | 7x7      |
| Convolutional | 1024    | 3x3         | 7x7      |
| Convolutional | 1000    | 1x1         | 7x7      |
| Avgpool     | Global  |             | 1000     |

Figure 5. Structure ResNet-18 network.
Step 4. Repeat Steps 2 and 3 until the clustering center do not change.

First, randomly select nine points as the cluster center, then calculate the intersection and union ratio of all other training data with the nine cluster centers. The cluster center with the largest intersection and union ratio is taken as the class of the training data. Then, the median of each class is calculated as the new clustering center. Continue to iterate until the cluster center does not change, and the final nine cluster centers are the final result. Figure 6 shows the final clustering result after 78 iteration times, where the red cross is the cluster center.

The width and height of the bounding box of each object are taken as the parameters to be clustered. The nine cluster centers in Figure 6 are [0.03163166, 0.0786631], [0.25076737, 0.47783422], [0.0430937, 0.1153877], [0.06252827, 0.16296791], [0.15647011, 0.31466578], [0.02147819, 0.05505348], [0.08710824, 0.12018717], [0.11113086, 0.19986631], and [0.05647011, 0.08312834]. The size of the input image is 416 × 416. The clustering result is the ratio of the width and height of the anchor box to the width and height of the original image. The parameters of the final anchor box should be transformed according to the original size, that is should be multiplied by 416. Then, the final nine cluster centers in the original input image are about [13, 33], [104, 199], [18, 48], [26, 68], [65, 131], [9, 23], [36, 50], [46, 83], and [24, 35]. The large anchor box is used to predict large objects and the small anchor box is used to predict small objects. Therefore, the three candidate boxes [46, 83], [65, 131], and [104, 1989] are used to predict bounding box on the scale of 52 × 52. Since the results of K-means clustering depend on the selection of initial clustering center, six clustering experiments were done in this article, as shown in Table 2.

The vehicle detection network optimized using feed-forward network and multi-scale clustering is as shown in Figure 7. The backbone network sends the extracted 13 × 13 features to the neck network, and the up-sampling multiplier of × 1, × 2, and × 4 to get 13 × 13, 26 × 26, and 52 × 52 size features, respectively, and then sends these different size features to different head networks to detect targets at different scales.

Experiments and results

To evaluate the performance of the optimized vehicle detection network, the accuracy $P$ and the recall $R$ are defined to evaluate the performance of the network. They can be calculated as

$$\begin{align*}
P &= \frac{TP}{TP + FP} \\
R &= \frac{TP}{TP + FN}
\end{align*} \quad (7)$$

where TP, FP, and FN denote the number of true positives, number of false positives, and number of false negatives, respectively. TP means that the vehicle on the road can be recognized and the classification is consistent with the actual type. FP means that the vehicle is recognized but the classification is not consistent with the actual type, or the background such as billboard is recognized as the vehicle. FN means that there is a vehicle in the image, but it is not recognized accurately.

Define the average accuracy as mAP, which denotes the area enclosed by the $P–R$ curve and the coordinate axis. It can be calculated as

$$mAP = \int_0^1 P(R) dR \quad (8)$$

The ideal value of mAP is 1. The closer of mAP value is to 1, the higher the detection accuracy of the model. The $P–R$ curves of the traditional and optimized vehicle detection network are shown in Figure 8.

Figure 8 shows that the mAP of the traditional YOLOv3-tiny network is 79.57%. After using the DarkNet-19 as the feedforward network to replace the feature extractors of the traditional YOLOv3-tiny network, the mAP is 82.87%. While using the ResNet-18 to replace the feature extractors of the traditional YOLOv3-tiny network, the mAP is 87.96%, which is better than the DarkNet-19. The proposed vehicle detection network optimized by adds multi-scale prediction to ResNet-18 as the feedforward network, the
mAP is 93.66%, which has an improvement of 14.09% compared with the traditional YOLOv3-tiny network.

Some of the results are shown in Figure 9. The green box in the figure shows the results of the traditional YOLOv3-tiny network, and the red box shows the results of the proposed vehicle detection network.

Figure 9 indicates that our improved method can show a high robustness to detect small object. The improved algorithm can detect small vehicles that cannot be detected by the original algorithm, and the bounding boxes generated by the improved algorithm are more accurate.
To evaluate the performance of the proposed vehicle detection network, several tests were conducted on the embedded platform TX2 and laptop. TX2 is an embedded artificial intelligence computing platform launched by NVIDIA in 2018 with Pascal architecture. GPU and CPU are consisted of two Denver and four A57 cores together. The laptop has a graphics card of NVIDIA GTX1060 6G and an Intel i7-7700HQ processor. The comparison result of each network is shown in Table 3.

Table 3 indicates that the traditional YOLOv3-tiny network has the simplest structure and the least number of parameters, and it can run at 24 fps on an embedded device. However, the detection accuracy is only 79.57%. While using the ResNet-18 to replace the...
feature extractors of the traditional YOLOv3-tiny network, it can run at speed on embedded devices with an accuracy of 87.96%, which is 8.39% higher than the traditional YOLOv3-tiny network. It can still reach 60 fps on the laptop. After adding multi-scale prediction to ResNet-18, the network runs at a similar speed and the size only increases by 2.6 MB. However, the detection accuracy improves by 5.7%. It can be seen that after two different levels of optimization, although the structure and the number of parameters becomes larger, and the speed of operation in the embedded hardware decreases. The detection accuracy improves from 79.57% to 93.66%, and it can reach 13 fps on embedded devices.

Conclusion

Vehicle detection methods based on deep learning show superiority over the traditional feature-based and machine learning-based methods. This article utilizes the traditional YOLOv3-tiny as the base network, proposed a light-weighted vehicle detection using the KITTI data set. Facing the poor vehicle detection accuracy of YOLOv3-tiny network, two light-weight models, DarkNet-19 and ResNet-18, were used to replace the original feature extraction network of YOLOv3-tiny. The detection accuracy increased from 79.57% to 87.96% when taking ResNet-18 to realize feature extraction. However, the detection of small objects is still poor, and the strategy of multi-scale is proposed. The K-means algorithm is utilized to cluster nine anchor boxes in three scales; they are $13 \times 13$, $26 \times 26$, and $52 \times 52$. Experimental results show that multi-scale strategy can effectively improve vehicle detection accuracy. The optimized vehicle detection light-weight network was tested on the embedded device TX2 and laptop. It was verified that the optimized network not only has improved detection accuracy, but also can meet the requirement of real-time.

In the follow-up work, we will further study the optimization of network and reduce time consumption. At the same time, more experiments and comparisons are helpful to reflect the superiority of our work. It is a huge challenge for application in real traffic environment; more data sets need to be supplemented to deal with the changeable objects under different situations.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is partially supported by the National Natural Science Foundation of China (grant nos 51975089 and 52175078) and the Natural Science Foundation Program of Liaoning Province (grant no. 2021-MS-127).

ORCID iD

Lie Guo https://orcid.org/0000-0001-6823-5488

References

1. National Bureau of Statistics of the People’s Republic of China. Annual statistics of the National Bureau of statistics of the People’s Republic of China in 2020, http://data.stats.gov.cn/.
2. Nguyen DT and Kajita Y. Traffic congestion and impact on the environment in Vietnam: development of public transport system—experience from actual operation of bus in Hanoi. *J Civil Environ Eng* 2018; 8: 317.
3. Zhu H, Yuen K, Mihaylova L, et al. Overview of environment perception for intelligent vehicles. *IEEE T Intell Transp* 2017; 18: 2584–2601.
4. Wang H, Yu Y, Cai Y, et al. Soft-weighted-average ensemble vehicle detection method based on single-stage and two-stage deep learning models. *IEEE Trans Int Veh* 2021; 6: 100–109.
5. Lee CH, Lim YC, Kwon S, et al. Stereo vision-based vehicle detection using a road feature and disparity histogram. *Opt Eng* 2011; 50: 027004.
6. Aqel S, Hmimid A, Sabri MA, et al. Road traffic: vehicle detection and classification. In: 2017 intelligent systems and computer vision (ISCV), Fez, Morocco, 17–19 April 2017, pp.1–5. New York: IEEE.
7. Noh S, Shim D and Jeon M. Adaptive sliding-window strategy for vehicle detection in highway environments. *IEEE T Intell Transp* 2016; 17: 323–335.
8. Nguyen H. Multiscale feature learning based on enhanced feature pyramid for vehicle detection. *Complexity* 2021; 2021: 5555121.

9. Zhang WW, Zheng Y, Gao Q, et al. Part-aware region proposal for vehicle detection in high occlusion environment. *IEEE Access* 2019; 7: 100383–100393.

10. Li QH, Garg S, Nie JT, et al. A highly efficient vehicle taillight detection approach based on deep learning. *IEEE T Intell Transp* 2021; 22(7): 4716–4726.

11. Chen S and Lin W. Embedded system real-time vehicle detection based on improved YOLO network. In: *2019 IEEE 3rd advanced information management, communicates, electronic and automation control conference (IMCEC)*, Chongqing, China, 11–13 October 2019, pp.1400–1403. New York: IEEE.

12. Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *2014 IEEE conference on computer vision and pattern recognition*, Columbus, OH, 23–28 June 2014, pp.580–587. New York: IEEE.

13. Redmon J, Divvala S, Girshick R, et al. You only look once: unified, real-time object detection. In: *2016 IEEE conference on computer vision and pattern recognition (CVPR)*, Las Vegas, NV, 27–30 June 2016, pp.779–788. New York: IEEE.

14. Liu W, Anguelov D, Erhan D, et al. SSD: single shot multibox detector. In: Leibe B, Matas J, Sebe N, et al. (eds) *Computer vision—ECCV 2016*. Cham: Springer International Publishing, 2016, pp.21–37.

15. Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans Pattern Anal* 2017; 39: 1137–1149.

16. Guo L, Wang D, Li L, et al. An accurate and fast single shot multibox detector. *IET Comput Vis* 2020; 14: 391–398.

17. Tajar AT, Ramazani A and Mansoorizadeh M. A lightweight tiny-YOLOv3 vehicle detection approach. *J Real-Time Image Pr* 2021; 18(6): 2389–2401.

18. Fu LS, Feng YL, Wu JZ, et al. Fast and accurate detection of kiwifruit in orchard using improved YOLOv3-tiny model. *Precis Agric* 2021; 22(3): 754–776.

19. Haq IU, Muhammad K, Hussain T, et al. Movie scene segmentation using object detection and set theory. *Int J Distr Sens N* 2019; 15: 1550147719845277.

20. Elgendy M, Sik-Lanyi C and Kelemen A. A novel marker detection system for people with visual impairment using the improved tiny-YOLOv3 model. *Comput Meth Prog B* 2021; 205: 106112.

21. Redmon J and Farhadi A. YOLOv3: an incremental improvement. *arXiv preprint arXiv* 2018; 1804.02767, https://arxiv.org/pdf/1804.02767.pdf

22. Song YN, Pan QK, Gao L, et al. Improved non-maximum suppression for object detection using harmony search algorithm. *Appl Soft Comput* 2019; 81: 105478.

23. Cheng R, He X, Zheng Z, et al. Multi-scale safety helmet detection based on SAS-YOLOv3-tiny. *Appl Sci* 2021; 11(8): 3652.

24. Geiger A, Lenz P and Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In: *2012 IEEE conference on computer vision and pattern recognition*, Providence, RI, 16–21 June 2012, pp.3354–3361. New York: IEEE.

25. Song H, Liang H, Li H, et al. Vision-based vehicle detection and counting system using deep learning in highway scenes. *Eur Transp Res Rev* 2019; 11: 51.

26. Wu Z, Sang J, Zhang Q, et al. Multi-scale vehicle detection for foreground-background class imbalance with improved YOLOv2. *Sensors* 2019; 19: 3336.

27. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: *2016 IEEE conference on computer vision and pattern recognition (CVPR)*, Las Vegas, NV, 27–30 June 2016, pp.770–778. New York: IEEE.

28. Cao Y, Wu Z and Shen C. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE T Circ Syst Vid* 2018; 28: 3174–3182.

29. Baruri R, Ghosh A, Banerjee R, et al. An empirical evaluation of k-means clustering technique and comparison. In: *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*, Faridabad, India, 14–16 February 2019, pp.470–475. New York: IEEE.