Front Obstacle Detection System based on YOLOv3

Chao Wang*, Yiyang Huang and Bowen Shi
Computer School, Beijing Information Science and Technology University

*Corresponding author e-mail: wangehao@bistu.edu.cn

Abstract: The task of front obstacle detection in unmanned driving is to allow the model to accurately detect types and corresponding position of each object. Complex true driving environment, multiple types of objects within range of visibility and multiple examples are challenges for detection. Particularly in unmanned driving field, not only the detection precision shall be pursued, but also the detection speed shall be guaranteed. This paper puts forward the front obstacle detection method based on YOLOv3. It establishes a Tensor Flow+Keras framework on GPU server cluster, designs and realizes YOLOv3, and realizes calibration and classification of front obstacles through training and optimization of the model on the data set. This method supports over 10 object types. Experimental results show validation set Loss (VAL Loss) reaches 21.206, mAP reaches 78.64, and detection speed reaches 30 FPS. The study provides theoretical basis for obstacles detection of automated vehicle in complex environment.

1. Introduction

In recent years, as the theoretical technology of artificial intelligence develops rapidly, unmanned driving, as a representative application in intelligent transportation field, has become the technical commanding point that technological giants try to seize. As a complex of sensor, Internet of Things, artificial intelligence and computer vision technology \cite{1}, the automated vehicles could recognize nearby environment, make decisions based on such information and realize autonomous action. The front obstacle detection system \cite{2}, like the eyes of automated vehicles, is the foundation of intelligent vehicles for environmental perception. An important link to be tackled in unmanned driving field is to accurately recognize front obstacles within limited time.

Zhao Zhengchun et al. \cite{3} realized front obstacle detection according to the principle of ultrasonic ranging so as to detect obstacle types which could not be detected accurately by low-cost method. Huang Wei et al. \cite{4} collected and processed nearby environment information with radar, image sensor and ultrasonic transducer, and analyzed target image by means of machine vision so as to reach obstacle detection effect. However, due to frequent missing or false detection caused by low algorithm accuracy, this method could not be applied to environment of true road surface. With improvement of image understanding ability of deep learning algorithm, CNN (Convolutional Neural Network) has been gradually applied to front obstacle detection. Shi Kaijing et al. \cite{5} proposed Faster RCNN-based vehicle detection method, and the detection speed on high-performance GPU (NVIDIA Geforce1080TI) may reach 0.2S/frame. Redmon J et al. \cite{6} put forward YOLO algorithm to further improve detection speed. Li Yunpeng et al. \cite{7} realized moving object detection in automatic driving based on YOLOv3; however, only three objects may be detected, respectively motor vehicles, pedestrians and riding. It is insufficient to recognize complex foregrounding information in driving environment, while it is unfavorable for vehicle driving.
For aforesaid problems, this paper adopts YOLOv3 algorithm (to get higher object detection speed) to distinguish small and big objects through multi-scale prediction. The clustering operation prior to training (based on K-means algorithm) improves precision of detection. In addition, this algorithm is applied to data sets with multiple categories and samples, and improves recognition capability of complex foregrounding information in driving environment.

2. YOLO algorithms

YOLO (you only look once) gathers position detection and image classification to one network, designs the bounding position to regressive parameters which could be acquired directly by network regression so as to greatly improve the speed. The design of YOLO algorithm is different from former deep learning algorithms applied to detection. The image processing of the latter algorithm is divided to links of candidate box extraction, convolution characteristics, SVM classification and bounding box regression. The characteristics extraction and classification process are carried out respectively, adding time complexity of algorithm.

The basic idea of YOLO is to get characteristic images of training data through Darknet. YOLO firstly divides characteristic images into S*S cells. Each cell shall predict B bounding-boxes and the confidence degree (probability of objects * IOU) and C types of prediction information; then get class-specific confidence score by multiplying confidence and prediction class information; filter bounding-box with low class-specific confidence score by setting threshold value, and finally get the most suitable bounding-box through restriction of non-maximum value and get the class information. The training process can be seen in Fig. 1.

Fig. 1 Calculation process of YOLO algorithm

In order to solve the defect of worse YOLO detection effect for small objects, YOLOv2 algorithm brings up the concept of Anchor-box. Anchor-box is the prior box size of K types of sizes. Each Anchor-box contains four offset values (x, y, w, h), i.e., center coordinate (x, y), width w and height h. Different sizes of objects will be predicted by anchor-box with different sizes, and then mapped to bounding-box.

In order to add precision of YOLOv2 detection, YOLOv3 adds network layers and multi-scale prediction. YOLOv3 network structure is divided to two parts. The first part is classification network, and the second is prediction network. The former adopts Darknet-53 network, with basic constitutional units of Convolutional and Residual. Convolutional takes charge of convolution, comprising Con2d, batch normalization and Leaky Relu; while Residual is of residual network structure. This layer is added to prevent vanishing gradient caused by information loss due to excessively deep layers. The prediction network extracts characteristic images through Darknet-53, and then detects objects with different sizes through multi-scale prediction.

YOLOv3 trains the network through reducing loss value of functions. The loss function of YOLOv3 is comprised of four parts, and the formula is shown below:
Formula (1) calculates the error between prediction coordinate \((x_i, y_i)\) and actual coordinate \((x_i', y_i')\); \(\lambda_{\text{coord}}\) represents weight coefficient set, and the error between actual width and length \((w_i, h_i)\) and predicted width and length \((w_i', h_i')\). Formula (2) calculates confidence loss. \(C_i\) is the true cell confidence and \(C_i'\) is the predicted cell confidence; in formula (3) \(p(c)\), \(p'(c)\) is respectively class prediction and actual class; then error function is generated. In the end, the three formulas are added to get the error function.

### 3. Design and realization of YOLOv3-based front obstacle detection program

Aiming at the study contents, this paper puts forward a YOLOv3-based front obstacle detection program. Relevant work is as follows: Firstly, apply pre-treatment to data set. Cluster data with K-means clustering algorithm to get prior box of Anchor-box (9 types). Train with YOLOv3-based network structure framework, and apply network optimization, including selecting different optimizers (adam), Stochastic gradient descent, SGD, and Adagrad, configuration and network model optimization of multiple parameters (batch-size, epoch, learning rate). Finally, carry out performance detection based on evaluation parameters to verify accuracy and detection speed of training model. The process can be seen in Fig. 2:

#### Dataset format conversion

- Bdd100k -> VOC

#### Anchor-box pre-initialization

- K-means

#### Training and tuning

- Adam/SGD/Adagrad

#### Evaluation

- mAP/ FPS

**Fig. 2** Process of YOLOv3-based front obstacle detection program

### 3.1 Pre-treatment of data set

The input image format received by YOLOv3 network is VOC, while Bdd100k data set is the document of json format (including label). It is necessary to convert format before training. The images are distinguished by name of json document label; attribute, timestamp and labels of images are in parallel with name. Attribute contains, weather, scene, and timeofday when the image is taken. Labels contain objects in images, and comprise category, box2d (coordinate on left lower corner and right upper corner of bounding-box), attribute and id. The task of data format conversion is to analyze each attribute of json document and fill in each document of xml; finally input xml into Annotations under VOC catalog.

### 3.2 Network parameters and optimizer

In network training, images are inputted to network for training. The size of each batch is called batch-size. Entire training of data is called one epoch; the times of one epoch iteration is total data divided by batch-size (if the batch-size is too large, the model is difficult to converge; if the batch-size is too small, the model is difficult to achieve expected accuracy). YOLOv3 architecture is based on Darknet53 which is commonly used in object detection network.
is too small, learning is slow, and batch-size is affected by training platform memory; the value shall be no larger than 9 in this paper). The larger the epoch value is, the more frequent the mode training will be and the better the model effect will be. However, the time spent in training will be longer. Moreover, when the model fits, it is meaningless to carry out epoch.

The network shall be optimized in order to reach the highest accuracy. In the initial training stage, the optimizer is usually selected to give initial weight to network. This paper adopts adam in training. As the training process deepens, when loss is no longer changed, the network configured by current optimizer has entered bottleneck stage. The optimizer shall be changed. SGD and Adagrad are selected in this paper for further optimization. SGD could adjust parameters manually, set learning rate and adjust learning step. Adagrad upgrades low-frequency parameters greatly while high-frequency parameters lessen. However, change in optimizer may unnecessarily bring active effect. Different adjustments shall be made according to specific situations as well. During the training process, the case that Loss is rising instead of lowering with SGD is found (see 3.3). That is because although SGD could reach the minimum value, it spends a longer time than other algorithms; additionally, it may be stuck in the saddle point. Adagrad is an optimizer suitable for sparse data. Bdd100k sample sets have multiple classes, and the gap between samples in different classes is large (see Section 3.2). Therefore, the training process configured by Adagrad will generate network model with higher robustness.

3.3 Anchor-box clustering pre-treatment based on K-means algorithm

Since the objects in front vision of automated vehicle are more, and usually contain more classes and examples of front obstacles. Before classifying various obstacles in image, YOLOv3 algorithm shall firstly box different objects in the image. Therefore, the initial candidate box set at random (the parameter is not optimal) will always affect speed of network convergence. K-means algorithm is used in this paper to get set parameter of Anchor-box of the training set. Experimental results show iteration frequency of network model after clustering required by convergence during training is reduced greatly. That is because K-means is a distance-based clustering algorithm. This algorithm introduces the concept of cluster. The cluster comprises objects with similar distance. Compact and independent clusters may be acquired through K-means algorithm. For given central point \((W_i, H_i), i \in \{1, 2, \ldots, K\}\) of K clusters, calculate distance \(d\) between every annotation box and center point of every cluster, \(d=1-\text{IOU (annotation box, cluster center)}\); distribute annotation box to the cluster center with the nearest “distance”; after different clusters are distributed to all annotation boxes, calculate clustering center point of every cluster, so as to get mean value of width and height of all annotation boxes in every cluster. The calculation formula can be seen in Formula 4,

\[
W_i = \frac{1}{N_i} \sum W_i, \quad H_i = \frac{1}{N_i} \sum h_i \quad \text{(Formula 4)}
\]

In the formula, \(N_i\) is the number of annotation boxes of the ith cluster.

Repeat aforesaid operation to get the set parameters of Anchor-box.

3.4 mAP performance detection algorithm

The definitions of mean average precision (mAP), precision and recall in this paper are as follows:

\[
\text{precision}(\text{class}) = \frac{\text{TP}}{\text{TP} + \text{TN}} \quad \text{Formula (5)}
\]

\[
\text{recall}(\text{class}) = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \text{Formula (6)}
\]

In the formula, TP refers to number of examples that detection model correctly recognizes classes to be detected; FP refers to number of examples with other detection classes as classes to be detected; FN refers to number of examples detecting classes to be detected as other classes. With Recall of certain class to be detected as x-coordinate, and precision of certain class to be detected as
y-coordinate, finally a P-R curve is drawn. The area of the curve is AP; get AP of every class, and the mean value is mAP.

4. Experiment and analysis

This paper, based on YOLOv3 network model, gets front obstacle detection model through training BDD100k dataset, evaluates model precision with mAP and tests detection speed with FPS.

4.1 Experimental environment

The experimental platform configuration can be seen in Table 1, including Intel corei7 processor, 16G RAM, 1TB hard disk, NVIDIA GeForce GTX graphics (configuring 8G video memory).

| GPU  | CPU   | Memory | Hard Disk capacity |
|------|-------|--------|--------------------|
| NVIDIA GTX 1070 | Intel corei7 | 16G | 1T |

4.2 Data set and pre-treatment of data set

The Bdd100k data set used in this paper is a large-scale automated driving data set released by Berkeley University. The data set contains 100,000 clips; the duration of each clip is 40s, resolution is 720p and frame rate is 30fps. The 10th second of every clip is sampled as the key frame to acquire 100,000 pieces of images (size: 1280*720), marked as json format label. The example of data set can be seen in Fig. 3.

![Fig. 3 Bdd100k data set](image)

There are 70,000 images of BDD100K training set, 20,000 images of test set, and 10,000 images of verification set. Totally ten label classes are contained, as shown in Table 2, covering all common individual elements in driving environment. The classes not only contain car, traffic light and traffic sign, but also contain bicycle (divided to rider and bike), person, motor and even train. The images of car are the most, totally 713211 pieces, the images of train are the least, totally 136 pieces. The difference of samples between different classes is large.

| Bdd100k detection classes and corresponding scale |
|-----------------------------------------------|
| class       | num       |
| traffic light | 186117    |
| traffic sign  | 239686    |
| car          | 713211    |
| person       | 91349     |
| bus          | 11672     |
| truck        | 29971     |
| rider        | 4517      |
| bike         | 7210      |
| motor        | 3002      |
| train        | 136       |
4. 3 Training process

For network optimization, the training process in this paper is divided to four stages. Configuration parameters and final results (VAL loss) in each stage can be seen in Table 3. In the first stage, the initial batch of training is 9, and epoch is 13; Adam is selected, and VAL Loss drops from 2000 to 28.17 and fluctuates around the value; in the second stage, the training batch is 8, epoch is 3; optimizers are respectively SGD and Adagrad; VAL Loss is rising instead of lowering when SGD is used, increasing to 36 from 28.17 at the end of the first stage and then lowering to 29. When the optimizer is Adagrad (learning rate is set to 0.001), the VAL Loss is increased to 31 and then dropped to 25, superior to that of SGD. Adagrad upgrades low-frequency parameters greatly while high-frequency parameters lesser. It is an optimizer suitable for sparse data set (as shown in Table 2, BDD100K training sets contain more classes, and gap between samples in different classes is large). During training process, Adagrad changes weight of car (713211 pieces) little, and weight of train (136 pieces) greatly; thus, it generates network model with higher robustness. In the third stage, Adagrad is used for training. The learning rate is set to 0.001, VAL Loss is dropped to 25; since change of Val Loss is little, this paper adjusts the learning rate to 0.001 to get the final effect of VAL Loss lowering to 21, so as to finish training.

| Table 3 BDD100K training process |
|----------------------------------|
|                                  |
| **Stage I**                      |
| **Stage II**                     |
| **Stage III**                    |
| batch                            |
| 9                                |
| 8                                |
| 5                                |
| epoch                            |
| 13                               |
| 3                                |
| 6                                |
| Optimizer                        |
| Adam                             |
| SGD                              |
| Adagrad                          |
| lr=0.01                          |
| lr=0.001                         |
| VAL Loss                         |
| 2000-28,17                       |
| 36-29                            |
| 31-25                            |
| 25-21                            |

4. 4 Train LOSS and VAL LOSS

The Train LOSS curve of YOLOv3 training model can be seen in Fig. 4. Totally 16 times of epoch are carried out in the training, so Train LOSS shows a descending trend as a whole; an obvious turning point is shown in the third epoch; the curve is relatively gentle in the 14th epoch, and the value is the lowest, 17.01, in the 16th epoch. The VAL LOSS curve of YOLOv3 training model can be seen in Fig.5, showing a descending trend according to times of epoch; an obvious turning point is shown in the second epoch; the curve is relatively gentle in the 11th epoch, and the value is the lowest, 21.206, in the 16th epoch.

![Fig. 4 Train LOSS curve](image-url)
4.5 mAP and FPS

The front obstacle detection model is used to detect x test set data. The mAP and FPS acquired can be seen in Table 4. mAP is, and FPS is 30fram/s; the detection effect can be seen in Fig. 6; the detection effects for car with larger image proportion and traffic sign with small proportion are favorable. To sum up, according to indexes of mAP and FPS, the model completely complies with real-time requirements.

| mAP  | FPS        |
|------|------------|
| 78.64| 30 Frame/S|

5. Conclusion

This paper adopts YOLOv3 to establish front obstacle detection model. This model may detect front obstacles with multiple sizes and classes. According to experiments, after the BDD100K dataset training model based on YOLOv3 is tested by verification set, Val loss may reach 21.206, mAP may reach 78.64 and FPS may reach 30 Frames/S. The future research directions could focus on optimization of YOLOv3 loss function; or future studies may add training epoch times, optimize training parameters, improve detection speed of the model, and apply the technology to actual unmanned driving.

6. Acknowledgements

This work is financially supported by the 2019 Student Research Training Project of BISTU (5101923400).
References

[1] Wang Zhiping. Study on automated vehicle and technical development [J]. Technology and Economic Guide, 2019,27(05):99.

[2] Wang Zhangu. Design of environment perception system for bus and research on detection method of the front obstacle [D]. Shandong University of Technology, 2018.

[3] Zhao Zhengchun, Lu Qirong, Jiang Dongchu. Design of barriers-detecting system for automobile [J]. Computer Measurement & Control, 2007(04):432-434.

[4] Huang Wei. Design and implementation of vehicle obstacle ahead detection system based on radar and computer vision [D]. Wuhan University of Technology, 2010.

[5] Shi Kaijing, Bao Hong, Xu Bingxin, Pan Weiguo, Zheng Ying. Preceding vehicles detection method of intelligent vehicles based on Faster RCNN [J]. Computer Engineering, 2018,44(07):36-41.

[6] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection[C], 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE Computer Society, 2016.

[7] Li Yunpeng, Hou Lingyan, Wang Chao. Moving object detection based on YOLOv3 during autonomous driving [J]. Computer Engineering and Design, 2019, 40 (04): 1139-1144.