Pedestrian detection based on one-stage YOLO algorithm

Xun Zuo¹, Jiaojun Li*¹, Jie Huang¹, Fan Yang¹, Tian Qiu¹ and Yang Jiang¹
¹Chongqing University of Technology, Chongqing, 400054, China
*Corresponding author’s e-mail: duj@cqut.edu.cn

Abstract. Pedestrian detection is mostly used in autonomous driving scenarios, which require high real-time detection. Most of the existing algorithms are based on Two-Stage detection, with poor real-time performance. This paper proposes a pedestrian detection system based on One-Stage, and uses One-Stage-based YOLO-Tiny, YOLO and YOLO-SPP algorithms to test and analyze the system in the scene of detecting pedestrians of different sizes. The results show that YOLO and YOLO-SPP have high average confidence, and YOLO-Tiny has a fast detection speed, which is suitable for fast detection scenarios.

1. Introduction

With the advancement of technology, the means of transportation have become more and more developed, and people’s travel has become more convenient, but at the same time it has also brought many safety hazards. Safety accidents frequently occur. The direct cause of this phenomenon is fatigue driving and drunk driving. The root cause is that the driver cannot accurately judge the road conditions in time and cannot take optimal measures in the event of an emergency. Therefore, in order to reduce the occurrence of traffic accidents, autonomous driving system has appeared, which uses the fast, stable and accurate computing power of computers to provide safe driving guarantee during the driving process of vehicles, which greatly reduces the incidence of traffic accidents.

In an autonomous driving system, pedestrian detection is a key point. It is necessary to detect not only the number and distance of the pedestrians in front, but also the specific location of the pedestrians, and provide reliable data for the central processor of the driving system to take measures to ensure the safe passage of vehicles and the safety of pedestrians.

Pedestrian detection belongs to the branch of object detection. Nowadays, object detection has become an increasingly popular direction. It can be widely used in various practical application fields such as industrial product detection, intelligent navigation, security monitoring, to help government agencies and the majority of enterprises improve work efficiency.

Currently, pedestrian detection programs are mainly divided into two categories. The first type of scheme is based on background modeling, find out the object of the foreground movement firstly, and then extract the features according to the specific area, and then use the classifier to classify to determine whether there are pedestrians, but this method mainly has the following problems.(1) The environment changes greatly, for example, the light will affect the chromaticity of the image.(2) When there are dense targets in the picture, the detection effect drops sharply.(3) The classifier must make a correct judgment on the change of the background object, but cannot classify it as a target object. The second type of scheme is based on statistical learning methods, which are currently commonly used methods for pedestrian detection. Pedestrian detection classifiers are constructed based on a large number of pedestrian samples prepared in advance and corresponding labels, which mainly represent deep learning. At present, deep learning has a large number of applications in the direction of pedestrian detection,
which is mainly divided into One-Stage and Two-Stage. Two-Stage is proposed before One-Stage detection. There are mainly algorithms that generate a series of samples and use them as candidate frames, and then use convolutional nerves. Network classification, the main representatives are R-CNN\([1]\), Fast-RCNN\([2]\), Faster R-CNN\([3]\). The One-Stage detection is a direct regression to predict the category and position coordinate information of the target. The accuracy is slightly lower than that of the One-Stage detection, but the speed is faster. It mainly represents the YOLO\([4]\) series and SSD\([5]\).

From traditional detection methods to deep learning applications, pedestrian detection has greatly improved in accuracy and computational complexity. From the representative of Two-Stage algorithm R-CNN, Fast-RCNN, Faster-RCNN to the representative of One-Stage algorithm SSD, YOLO, and even SqueezeNet\([6]\) in the recent two years, whether in accuracy, Real-time or model lightweight, there have been breakthroughs. Before the appearance of SSD and YOLO algorithms, although the accuracy of target detection was considerable, the real-time performance could not meet the requirements. As shown in Table 1, the experimental results of different methods on the PASCAL_VOC data set.

| model      | class    | data set | network    | mAP  | FPS |
|------------|----------|----------|------------|------|-----|
| R-CNN      | Two-Stage| VOC 2007 | VGG16      | 66.0 | —   |
| Fast R-CNN | Two-Stage| VOC 2007 | VGG16      | 70.0 | 3   |
| Faster R-CNN| Two-Stage| VOC 2007 | ResNet101 | 76.4 | 7   |
| YOLO       | One-Stage| VOC 2007 | VGG16      | 66.4 | 45  |
| SSD        | One-Stage| VOC 2007 | VGG16      | 75.1 | 23  |

Aiming at the problem of poor real-time detection of Two-Stage, this paper proposes a pedestrian detection system based on One-Stage, and uses One-Stage-based YOLO-Tiny, YOLO and YOLO-SPP algorithms to detect pedestrians of different sizes to test system. The results show that YOLO and YOLO-SPP have high average confidence, and YOLO-Tiny has a fast detection speed, which is suitable for fast detection scenarios.

2. Pedestrian detection system model

This paper implements pedestrian detection based on the YOLO algorithm. The YOLO algorithm creatively predicts \(n\) boxes in each area on the area where the feature map is finally output \(7 \times 7\). These boxes have different sizes and positions, covering almost all possible target positions. And predict the offset of the preselection box and the real object, rather than the absolute position coordinates, as in equation (1).

\[
\begin{align*}
    b_x &= \sigma(t_x) + c_x \\
    b_y &= \sigma(t_y) + c_y \\
    b_w &= p_w e^{w_x} \\
    b_h &= p_h e^{h_x}
\end{align*}
\]

\(c_x\) and \(c_y\) represents the coordinates of the upper left corner of the area where the target is located. \(\sigma(t_x)\) and \(\sigma(t_y)\) represents the normalized distance between the center point of the prediction box and the upper left corner of the target area. \(p_w\) and \(p_h\) represents the width and height of the current preselection box. And use the sum of different weights as the loss function for the boxes position loss, confidence loss and category loss, as in equation (2).
In addition, YOLO-Tiny, YOLO-SPP algorithm and YOLO are used to compare the detection accuracy and speed. YOLO-Tiny is a tailored version of YOLO on a convolutional network, and YOLO-SPP is a fusion version of YOLO and SPP-Net\cite{7} algorithms.

This paper uses the deep learning framework PyTorch to build the overall target detection network structure of YOLO. Firstly, use LabellImg software to manually label the pedestrian data set. Through the built-in library functions provided by PyTorch, analyze the three types of network structure information cfg files and build a deep learning model. Secondly, perform preprocessing operations such as data enhancement on the picture, Send it to the network in batches for forward propagation and back propagation training, and finally use the pre-training weights to obtain a new weight file, and use the interface DNN provided by OpenCV to analyze the deep learning model, load the trained model, and input the image Test with the built-in camera of the laptop. The overall design framework is shown in Figure 1.

![System overall design framework.](image)

Figure 1. System overall design framework.

3. Pedestrian detection experiment and data analysis

This section compares the vertical and horizontal tests of this article on a notebook computer with an i5-6300HQ CPU and GTX960M GPU, that is, input to the YOLO, YOLO-Tiny, and YOLO-SPP networks when the CPU and GPU acceleration are used separately. There are Picture and camera test. The picture test is divided into small object and large object test, and the test results are compared for accuracy and test speed. Specifically, the DNN deep learning module provided by OpenCV is used to test the trained model loading.

3.1. Picture test

In this section, the model will be tested on a single picture. First, use a small target picture to test and observe the detection effect when the target is small in the picture. Then use the big target picture to test and observe the detection effect when the target takes a large proportion in the picture.
3.1.1. Small object test. The detection of small targets mainly relies on shallow feature maps. The shallow feature maps have smaller receptive fields than deep feature maps, which is conducive to the detection of small object. This is due to the network structure of YOLO, which is not as effective as other networks for small object detection. YOLO increases the fusion of shallow, middle, and deep feature maps, and improves the ability to detect small object.

![Test results](image)

Figure 2. Small object test results.

The test picture is a picture with a relatively small proportion of pedestrians in the entire image, and other objects such as the background have large interference. The picture pixels are $800 \times 600$. Three different networks are used for detection. The detection results are shown in Figure 2. (a) is the YOLO test result, (b) is the YOLO-Tiny test result, (c) is the YOLO-SPP test result, there are 5 person in the figure. As shown in the figure, under the CPU operation, the detection time using the YOLO model is 1.405s, and 5 person objects are detected. Under the premise that the recall rate is about 100%, the average confidence is 95.2%; the detection time using the YOLO-Tiny model is 0.233s, 3 person objects are detected. Under the premise that the recall rate is about 60%, the average confidence is 68%; the detection time using the YOLO-SPP model test is 1.782s, and 4 person objects are detected. The recall rate is about Under the premise of 80%, the average confidence is 84%. Under GPU acceleration, the detection times of the three models are 0.094s, 0.031s, and 0.106s, respectively, an increase of about 1394.1%, 651.6%, and 1578.6% year-on-year. Table 2 Comparison of test indicators for each model.

| model    | pixel    | time consumed (CPU) | time consumed (GPU) | recall | Average confidence |
|----------|----------|---------------------|---------------------|--------|--------------------|
| YOLO     | 800 × 600 | 1.405s              | 0.094s              | 100%   | 95.2%              |
| YOLO-Tiny| 800 × 600 | 0.233s              | 0.031s              | 60%    | 68%                |
| YOLO-SPP | 800 × 600 | 1.782s              | 0.106s              | 80%    | 84%                |

3.1.2. Large object test. Large object are mainly detected through deep feature maps. The deep feature maps have the characteristics of small window, large perception field and many channels.
Figure 3. Large object test results.

The test picture is a picture with a large proportion of pedestrians in the whole picture, and other targets such as the background have little interference. The selected picture pixels are $800 \times 600$, using three different networks respectively. The detection results are shown in Figure 3, (a) is YOLO Test results, (b) are the YOLO-Tiny test results, (c) are the YOLO-SPP test results. It can be seen from the detection effect diagram that there are a total of 5 person objects in the figure. Under the CPU operation, the detection time using the YOLO model is 1.383s, and 4 person objects are successfully detected. Under the premise of a recall rate of 80%, the average confidence The detection time using the YOLO-Tiny model is 0.207s, and 4 person objects are successfully detected. Under the premise that the recall rate is 80%, the average confidence is about 69.5%; the detection using the YOLO-SPP model test The time is 1.539s, 4 person objects are successfully detected, and the average confidence is 91% under the premise of a recall rate of 80%. Under GPU acceleration, the detection times of the three models were 0.107s, 0.029s, and 0.105s, respectively, which increased by 1192.7%, 613.9%, and 1365.2% year-on-year. Table 3 Comparison of test indicators for each model.

| model          | pixel      | time consumed (CPU) | time consumed (GPU) | recall | Average confidence |
|----------------|------------|---------------------|---------------------|-------|--------------------|
| YOLO           | 800 × 600  | 1.383s              | 0.107s              | 80%   | 94.25%             |
| YOLO-Tiny      | 800 × 600  | 0.207s              | 0.029s              | 80%   | 69.5%              |
| YOLO-SPP       | 800 × 600  | 1.539s              | 0.105s              | 80%   | 91%                |

3.2 Real-time camera test

In practical applications, intelligent security, automatic driving and other fields need to use pedestrian detection, especially automatic driving, which has extremely high requirements for the speed and real-time performance of pedestrian detection. Therefore, this section will discuss the real-time performance
of the detector. The test is performed through the built-in camera of the laptop. Since the built-in camera is a VGA camera, the maximum pixel is $640 \times 480$, and because the foreground and background in the image cannot be exactly the same when the real-time camera is turned on, the test results have certain deviations. The test result is shown in Figure 4, (a) is the YOLO test result, (b) is the YOLO-Tiny test result, (c) is the YOLO-SPP test result.

![Image](a) ![Image](b) ![Image](c)

Figure 4. Real-time camera test results.

It can be seen from the test result graph that there is only one person object, all three networks have detected the target, and the recall rate reaches 100%, but the confidence and test speed of the three are different. Under CPU operation, the confidence level of using the YOLO network is 94%. Take the detection time of ten consecutive frames and calculate the average time. The average value is about 1.285s, and the number of frames is about 0.78 frames. The picture is obviously stuck and cannot meet real-time Inspection task. The confidence level of using the YOLO-Tiny network is 43%, the average detection time is about 0.202s, and the number of frames is about 4.95 frames, which barely meets the real-time detection task. The confidence of using the YOLO-SPP network is 90%, the average detection time is about 1.347s, and the number of frames is about 0.74, which cannot meet the real-time detection task. Under GPU acceleration, we also take ten consecutive frames to calculate the average detection time. The average detection time of the three models is about 0.087s, 0.026s, 0.086s, and the number of frames is about 11.49, 38.46, and 11.63, respectively. The number of frames increased by about 1373.0%, 677.0%, and 1471.6% respectively. The indicators of the three models for real-time camera testing are shown in Table 4.

| model     | pixel     | time consumed (CPU) | time consumed (GPU) | recall | Average confidence |
|-----------|-----------|---------------------|---------------------|--------|--------------------|
| YOLO      | $640 \times 480$ | 1.285s             | 0.087s             | 100%   | 94%                |
| YOLO-Tiny | $640 \times 480$ | 0.202s             | 0.026s             | 100%   | 43%                |
| YOLO-SPP  | $640 \times 480$ | 1.347s             | 0.086s             | 100%   | 90%                |
4. Conclusion
This paper uses the Pytorch framework to compare horizontally and vertically in pedestrian detection with YOLO, YOLO-Tiny, and YOLO-SPP. YOLO and YOLO-SPP networks can guarantee a high recall rate, the average confidence can reach more than 80%, using the CPU to process each picture is usually more than 1s, mainly because YOLO and YOLO-SPP use complete Compared with YOLO-Tiny's YOLO network structure, the amount of calculation is much larger, but under GPU acceleration, the number of frames can reach about 10 frames, which can barely meet the real-time performance, and the effect is improved significantly. Using YOLO-Tiny as the input network is slightly inferior to the other two networks in terms of recall and confidence. However, due to the lighter network structure of YOLO-Tiny, the amount of calculation is small when performing detection tasks. The processing speed of pictures and videos is faster, and it is better than the other two networks in real-time.

References
[1] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. IEEE, 2014: 580-587.
[2] Girshick R. Fast R-CNN[J]. Computer ence, 2015.
[3] Ren S Q, He K M, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks[J].IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(6): 1137-1149.
[4] Redmon J, Divvala S, Girshick R, et al. You only look once: unified, real-time object detection[C]//Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition(CVPR).Las Vegas, NV, USA:IEEE, 2016: 779-788.
[5] Liu W, Anguelov D, Erhan D, et al. SSD: single shot multibox detector[C]//Proceedings of the 14th European Conference on Computer Vision (ECCV). Amsterdam, Netherlands: Springer,2016:21-37.
[6] IANDOLA F N, HAN S, MOSKEWICZ M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size[C] // 5th International Conference on Learning Representations, Toulon, Fr, 2016: 1-13.
[7] He K, Zhang X, Ren S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 37(9):1904-1916.