Academic Journal of Science and Technology
ISSN: 2771-3032 | Vol. 5, No. 1, 2023

Vehicle Object Detection Based on Deep Learning

Zhaoming Zhou, Hui Li

School of Mechatronic Engineering, Southwest Petroleum University, Chengdu 610500, China

Abstract: With the continuous improvement of science and technology and living standards, cars have become a necessary means of transportation for people to travel, which is bound to be followed by traffic accidents, so it is particularly necessary to detect or identify cars. Aiming at the above problems, this paper focuses on the application of deep learning in vehicle recognition problems, and is committed to finding a vehicle recognition algorithm with high recognition rate and stability. This paper focuses on the analysis of SSD algorithm and its basic theory convolutional neural network. Finally, this paper uses the pictures of stationary vehicles and a film video respectively to identify the stationary and moving state of vehicles. In the process of image recognition, the rationality and accuracy of the proposed method are verified according to the accuracy rate given above the image.

Keywords: Deep learning, Vehicle detection, OpenCV, SSD algorithm.

1. Introduction

With the continuous development of science and technology, cars have become an indispensable means of transportation in People's Daily life, which is also the fundamental cause of traffic congestion and a necessary factor for traffic accidents. It is the existence of these two situations that increase the work burden of traffic police and related units. Beyond that, tracking vehicles is the most immediate problem facing government agencies. In order to alleviate this burden and even eliminate all kinds of possible problems, necessary intelligent monitoring of vehicles has become an effective means. Deep learning not only solves these problems, but is now being applied to every aspect of life. Deep learning was first proposed by Hinton[1] in 2006. Since then, it has played an irreplaceable role in People's Daily life. Therefore, scholars at home and abroad have conducted extensive research on the theoretical work and application objects of deep learning. Hayit Greenspan et al.[2] take into account the advanced nature of deep learning and realize the power of Convolutional Neural Network (CNN) in processing visual tasks, so they use CNN to study the positioning work of deep learning in medical image processing. It also provides some help for the future medical image processing and the development of medical career. Amodei D et al.[3] replaced the entire pipeline of hand-designed components with neural networks, and end-to-end learning enabled us to process a wide variety of speech, including noisy environments, accents, and different languages. Finally, this method is used to recognize two distinct speech sounds of English and Mandarin, which opens a successful door for future speech recognition work. Finally, looking at the advantages of deep learning in natural language processing, American scholar Ronan Collobert et al.[4] described a single convolutional neural network architecture that, given a sentence, outputs a series of language processing predictions: part of speech tags, blocks, named entity tags, semantic roles, the likelihood that semantically similar words and sentences will have meaning. The entire network uses re-sharing (instances of multitasking learning) to train all of these tasks together. Domestic research on deep learning has also made great progress and obtained a lot of research results. Ding Hong and Rao Wanxian[5] used the deep belief network to detect human behavior. Before using the deep belief network for recognition, the author preprocessed the data, denoised the data using wavelet technology, and then used PCA method for main frequency analysis. Finally, the processed data was substituted into the deep belief network for training, and the model was obtained for human behavior recognition. The deep belief network proposed in this paper is as effective as other machine learning methods. In order to improve the image recognition accuracy, Gao Yuan et al.[6] proposed a wide residual super-resolution neural network based on depth-detachability convolution. This method divides the channels of the convolutional layer into several groups and normizes the data of each group.

Based on the subject of vehicle target detection, this paper selects OpenCV toolbox[7] and uses the method of single target multi-box detection (Single Shot MultiBox Detector,SSD)[8,9]combined with deep neural network (Deep Neural Network, DNN)[10,11]to achieve accurate and efficient vehicle positioning and recognition.

2. The Theoretical Basis of The Research

In this paper, MobileNet SSD in OpenCV and deep neural network module are combined to construct vehicle target detector.

2.1. Deep neural network DNN

DNN divides neural network layers into three categories according to the layers in different positions, namely input layer, hidden layer and output layer. Fig. 1 shows the specific figure.

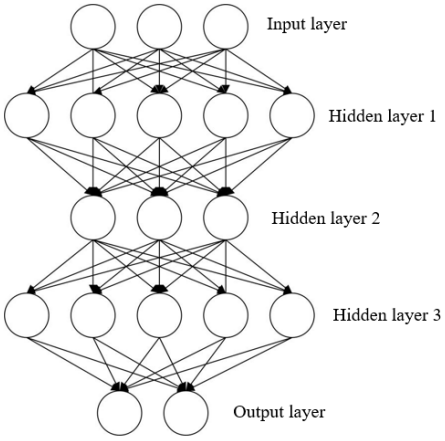


Figure 1. Neural network structure

The middle layer and layer of DNN can be fully connected. Specifically, the neurons in layer i are connected with each neuron in layer $i+1$, and the specific connection mode can be expressed as Equation (1).

$$z = \sum w_i x_i + b \tag{1}$$

In equation (1), z represents the output of neurons in layer $i+1$, w_i represents the weight vector of neurons in layer i, x_i represents the value of neurons in layer i, and b represents the bias of neurons in Layer i.

The definition of parameters between neurons in DNN can be shown in Fig. 2.

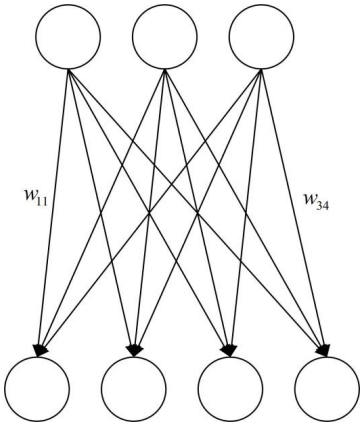


Figure 2. Stealth relationship between the two layers of neurons

It can be seen from the figure that the connection between the two layers of neurons is connected by weight. In the figure, w_{11} represents the weight of the connection between the first neuron of the first layer and the first neuron of the next layer, and w_{34} represents the weight of the connection between the third neuron of the first layer and the fourth neuron of the next layer. Take the output of the first neuron in the second layer for example, as shown in Equation (2).

$$z = w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + b \tag{2}$$

In Formula (2), x_1, x_2 and x_3 respectively represent the values of the three neurons in the first layer, b represents the bias of the first layer, and b represents the weight. The forward propagation of eigenvalues in a neural network is that multiple neurons in Fig. 2 are overlapped until the final output.

2.2. Convolutional neural networks CNN

The classic CNN model is shown in Fig. 3, in which 2D images are directly input into the network and then convolved with several adjustable convolution kernels to generate corresponding feature maps to form layer C1. The feature map in layer C1 will be sampled twice to reduce its size and form layer S1. Typically, the pool size during mapping is 2×2. This process will be repeated in layers C2 and S2. After sufficient features are extracted, the 2D pixel raster is converted into 1D data and input into the traditional neural network classifier.

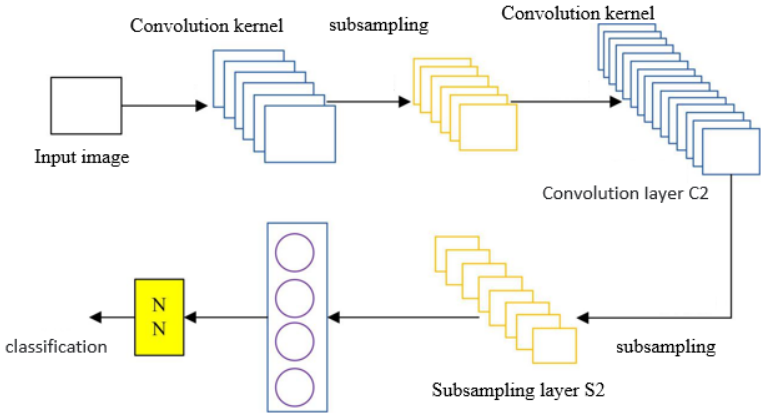


Figure 3. Convolutional neural network structure

Entering the convolution layer, the upper layer feature map is divided into many local regions and convolved with trainable kernels respectively. After processing convolution by activation function, we will obtain new output feature maps. Let the l layer be the convolutional layer, and the output of j layer can be expressed as:

$$X_j^l = f(\sum_{i \in M_j} X_i^{l-1} * K_{ij}^l + b_j^l) \quad (3)$$

In equation (3),M_j represents the local region connected by the j kernel,K_{ij}^l is a parameter of the convolution kernel, K_{ij}^l is a bias, $f()$is a Gaussian kernel.

In the subsampling layer, the most common method is the mean pool in the 2×2 region. That is, an average of 4 points in the area serve as new pixel values.

Parameter estimation in CNN still uses the gradient algorithm of back propagation. However, according to the characteristics of CNN, we should make some modifications in a few specific steps.

Here, assume that the residual error vector propagated to the raster layer is. Specifically, it can be expressed as formula (4).

$$d_r = [d_{111}, d_{112}, \dots, d_{jmn}]^T \quad (4)$$

Since rasterization is a 2D-to-1D transmission, it is only necessary to reorganize the residual error vector from the 1D to 2D matrix and pass it back to the subsampling layer.

When inverting from S layer to C layer, different pooling methods correspond to different processes of back propagation of residual error. In the average convergence, we simply average the residual error of the current point to the top 4 points. Here, it is recommended that the residuals at the point of layer S be Δ_q . After subsampling, the error transmitted to layer C can be expressed as Equation (5).

$$\Delta_p = upsample(\Delta_q) \quad (5)$$

Next, the trainable parameters in layer C are analyzed. Layer C has two tasks in backpropagation: reverse the residuals and update their parameters. According to BP algorithm and considering convolution operation, we can get the formula for updating parameter θ_p in the convolution layer as follows:

$$\frac{\partial E}{\partial \theta_p} = rot180((\sum X_q) * rot180(\Delta p)) \quad (6)$$

In equation，$rot180$ represents a 180 degree rotation of the matrix.

If the feature mapping in the former S layer q' is connected to the set C in the convolution layer p , the extended residual error q' can be expressed as:

$$\Delta_{q'} = (\sum_{p \in C} \Delta_p * rot180(\Delta p)) X_{q'} \quad (7)$$

After updating all parameters, the network completes a round of training, which should be performed for all training samples until the entire network meets the training requirements.

2.3. Principle of SSD algorithm

The SSD algorithm is a branch algorithm of CNN. SSD convolutional neural network is a supervised deep learning model, which is mainly composed of three layers: convolutional layer, activation function layer and pooling layer. In a standard SSD (SSD300), 8732 bounding boxes and 8732 scores per category are generated for each input image. The output after the non-maximum suppression step is the final detection result. Non-maximum suppression is an algorithm that attempts to eliminate additional boundary boxes based on scores and intersections. In other words, non-maximum suppression attempts to merge all bounds suggested as unique objects. In the early layers of SSDS, a standard architecture called the underlying network was used for high-quality image representation (truncated before any classification layer).

3. Vehicle Object Detection Based on SSD

3.1. Object detection analysis based on OpenCV

This chapter combines MobileNet SSD and DNN module in OpenCV to build deep learning object detector. This article introduces the code in several parts. First, create anew project file in Python and write code to import some of the necessary toolkits in OpenCV, as shown in Fig. 4.

```
import numpy as np
import argparse
import cv2
i=0
# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=False,default='./image.jpg',
 help="path to input image")
ap.add_argument("-p", "--prototxt", required=False,default='./MobileNetSSD_deploy.prototxt',
 help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=False,default='./MobileNetSSD_deploy.caffemodel',
 help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.4,
 help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
```

Figure 4. Importing and setting the tool package

Set the necessary Settings for target detection. The setting methods are manual and automatic. --image indicates setting

the path to the input image, -- prototxt indicates setting the location of the Caffe file, --model indicates setting the location of the pre-training model, and --confidence indicates setting the minimum probability threshold of the filter weak signal detector, the default value being 20%.

After completing this series of preparatory work, set class labels and detection box colors for detection targets, as shown in Fig. 5.

```
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
  "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
  "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
  "sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
```

Figure 5. Setting labels and colors

As shown in Fig. 5, it is not difficult to find that the class label set by this program includes two categories of "bus" and "car" to be studied, which can be accurately detected, and then the border color of the object to be detected is set. After setting relevant parameters, the model needed for analysis in this paper needs to be loaded. The program of loading the model is shown in Fig. 6.

```
# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# load the input image and construct an input blob for the image
# by resizing to a fixed 300x300 pixels and then normalizing it
```

Figure 6. Loading the model

After loading the model, you need to load the query image and perform blob analysis. The blob feeds forward through the network, as shown in Fig. 7. blob analysis is the analysis of the connected domain of the same pixels in the image, which is called blob. Blob analysis tools can isolate objects from the background and calculate the number, location, shape, orientation, and size of objects, as well as provide a topology between related spots.

```
# (note: normalization is done via the authors of the MobileNet SSD
# implementation)

image = cv2.imread(args["image"])
(h, w) = image.shape[:2]
blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 0.007843, (300, 300), 127.5)
```

Figure 7. Feature extraction

In Fig. 7, the image is first loaded, then the height and width of the image are extracted, and a blob of 300×300 pixels is calculated from the image. After feature extraction is completed, the process of feature transmission is followed, as shown in Fig. 8.

```
# pass the blob through the network and obtain the detections and
# predictions
print("[INFO] computing object detections...")
net.setInput(blob)
detections = net.forward()
# loop over the detections
```

Figure 8. Setting the forwarding network

In Fig. 8, we need to set the network input, calculate the forward transmission result of the input and store the result as detection. The running time of the program here is closely related to the size of the input features and the model chosen, but the method chosen in this paper can be implemented on ordinary CPU without GPU intervention. After all configurations are complete, you need to implement cyclic detection and determine the location of the detected object. Fig. 9 shows the detection process.

```
# prediction
confidence = detections[0, 0, i, 2]
# filter out weak detections by ensuring the `confidence` is
# greater than the minimum confidence
if confidence > args["confidence"]:
# extract the index of the class label from the `detections`,
# then compute the (x, y)-coordinates of the bounding box for
# the object
    idx = int(detections[0, 0, i, 1])
    if idx != 7:
        continue
box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])

(startX, startY, endX, endY) = box.astype("int")

# display the prediction
label = "{}: {:.2f}%".format(CLASSES[idx], confidence * 100)

print("[INFO] {}".format(label))

cv2.rectangle(image, (startX, startY), (endX, endY),
COLORS[idx], 2)

y = startY - 15 if startY - 15 > 15 else startY + 15

cv2.putText(image, label, (startX, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
```

Figure 9. Target detection

In Fig. 9 we first loop through our detection, remembering that multiple objects can be detected in a single image, in addition to checking the confidence (i.e., probability) associated with each detection. If the confidence level is high enough (that is, above the threshold), then we display the prediction in the terminal and draw the prediction on the image using text and color bounding boxes. The specific detection steps are as follows:

Step p1: Loop test the object to be tested, and extract the confidence value using the confidence function;

Step2: Judge the size of the confidence value. If the confidence value is greater than the minimum threshold set, extract the class index label and calculate the boundary box around the detected object;

Step3: After calculating the value of the boundary box, extract the coordinate value of the border, and use it to draw a rectangle and display text;

Step4: The display text needs to be constructed by itself, so next it needs to build a text label containing the class name and confidence;

Step5: Use the label, print it to the terminal for display, and then use the coordinates extracted before to draw a color rectangle around the object;

Step6: Set the label display position, which is not important in the whole detection process. Under normal circumstances, we want the label to be displayed above the rectangle, but if there is no space, we will display it below the top of the rectangle. The setting here can be set according to the preferences of scholars.

Step7: Finally, use the value just calculated to cover the color text on the image.

After all the detection steps are completed, the detected images need to be displayed to the readers, as shown in Fig. 10.

```
# show the output image
cv2.imshow("Output", image)
cv2.imwrite("output.jpg",image)
cv2.waitKey(0)
```

Figure 10. Detection result

So far, the OpencV-based target detection system has completed all the functions of static detection. For the dynamic detection of detection objects, it only needs to change image to video before the program. The target detection results under the above two states will be shown in the next chapter.

3.2. Vehicle detection analysis based on OpenCV

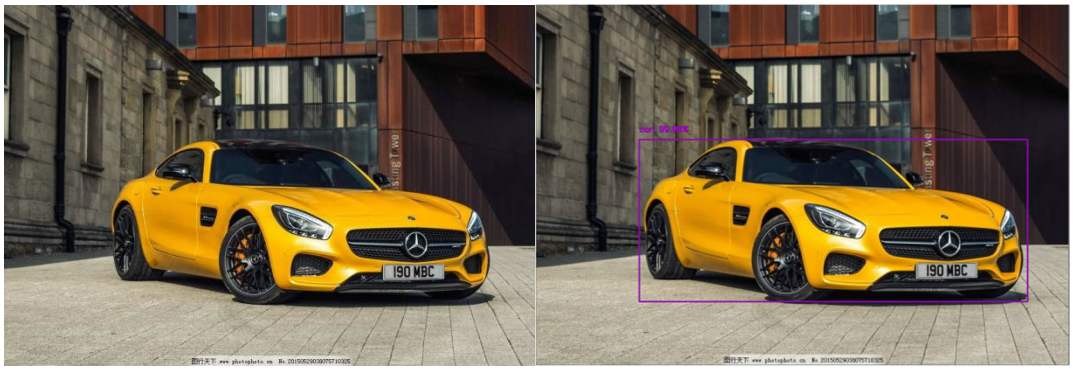
In general, the vehicle state is divided into two states: moving and stationary. A relatively complete vehicle recognition system based on computer vision consists of five parts: vehicle image information acquisition, vehicle image preprocessing, vehicle image segmentation, vehicle image feature extraction, vehicle recognition. The above 5 parts are equally applicable to both moving and stationary vehicles. Vehicle image information can be captured by high-speed camera or video data stream collected by high-definition camera. After the acquisition of image information, it is necessary to preprocess the image information. The processing content includes information denoising processing and image effect enhancement processing. After the

completion of image preprocessing, the image needs to be segmented. The purpose of image segmentation is to determine whether there is a vehicle in it. If there is a vehicle, the next step of feature extraction can be carried out. Classification processing mainly includes establishing vehicle classification model according to vehicle sample information, and adjusting relevant parameters in the model, and finally realizing the whole vehicle identification work based on this.

4. The Case Analysis of Vehicle Detection

4.1. Static vehicle image recognition

Firstly, the vehicle image is detected according to the requirements of this paper. The original image and detection results are shown in Fig. 11.



(a) Original drawing of vehicle (b) Vehicle test result

Figure 11. Comparison of vehicle detection

The vehicle in Fig. 11(a) is detected, and the results are shown in Fig. 11(b). It can be seen that the detection algorithm correctly gives the frame where the car is located, and the detection accuracy rate is as high as 99.99%. In order to exclude accidental circumstances, this paper will give a comparison diagram of another group of test results, as shown in Fig. 12.

(a) Original drawing of vehicle (b) Vehicle test result

Figure 12. Comparison diagram of vehicle detection

In order to show that the detection result in Fig. 11 is not accidental, some other background is added to the material given in Fig. 12, and the location of the vehicle is very far away. However, according to the detection results, this method can well identify the vehicle information, and it can be seen from Fig. 12 that the accuracy rate of vehicle recognition is up to 99.43%. From the point of view of probability theory we can say that this is a car. The above are the detection results of a single target. In order to illustrate the correctness of the method, the detection results of multiple targets are shown in Fig. 13.

(a) Original drawing of vehicle (b) Vehicle test result

Figure 13. Comparison of vehicle detection results

Fig. 13 shows that the detection method proposed in this paper can detect not only a single vehicle, but also multiple vehicles. The results are shown in Fig. 13(b), where the detection accuracy of two vehicles is 99.99% and 97.73% respectively, which can also illustrate the accuracy of the results.

4.2. Dynamic vehicle image recognition

For dynamic vehicle image recognition, this paper captures a film video and identifies the vehicles in it. The recognition process is shown in Fig. 14.

(a) Dynamic Vehicle identification I (b) Dynamic vehicle identification II

(c) Dynamic vehicle identification III (d) Dynamic vehicle identification IV

(e) Dynamic Vehicle identification V (f) Dynamic vehicle identification VI

Figure 14. Dynamic vehicle identification diagram

Fig. 14 shows the vehicle recognition process in the film material provided in this paper. The six subgraphs show the appearance of cars as the result of recognition respectively. It can be seen that the algorithm provided in this paper can also better realize the recognition of the whole vehicle in the process of vehicle movement. The figure not only contains the recognition of a single vehicle, but also includes the recognition of multiple vehicles. The figure also includes the recognition of vehicles in good light and dim light. No matter what kind of environment, the recognition method in this paper can better identify the dynamic process of vehicles.

5. Conclusion

This paper takes vehicle recognition as the research objective, and studies the application of deep learning in image processing, which includes image information acquisition, vehicle image preprocessing, vehicle image segmentation, vehicle image feature extraction and vehicle recognition several processes. In the research process, the deep learning method of convolutional neural network is studied, and the SSD algorithm based on convolutional neural network is used to realize the recognition process of vehicle image. In the specific process of identification, OpenCV, an open source tool box, is used to detect the target object, and the detection of stationary state and moving state of the vehicle is realized.

References

[1] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets[J]. Neural computation, 2006, 18(7): 1527-1554.

[2] Greenspan H, Van Ginneken B, Summers R M. Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique[J]. IEEE Transactions on Medical Imaging, 2016, 35(5): 1153-1159.

[3] Amodei D, Ananthanarayanan S, Anubhai R, et al. Deep speech 2: End-to-end speech recognition in english and mandarin[C]. International conference on machine learning. 2016: 173-182.

[4] Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning[C]. Proceedings of the 25th international conference on Machine learning. ACM, 2008: 160-167.

[5] Ding Hong, Rao Wanxian. Design of Human Behavior Detection System based on Deep Learning [J]. Electronic Technology and Software Engineering,2019,No.155(09):79-80.

[6] Gao Yuan, Wang Xiaochen, Qin Pinle, et al. Superresolution reconstruction of medical image based on depth-separable Convolution and wide residual Network [J]. Journal of Computer Applications,2019,39(09):2731-2737.

[7] Jibbe M K, Kannan S. Method to handle demand based dynamic cache allocation between SSD and RAID cache: U.S. Patent Application 12/070,531[P]. 2009-8-20.

[8] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]. European conference on computer vision. Springer, Cham, 2016: 21-37.

[9] Qin Xiaowen, Wen Zhifang, Qiao Weiwei. Image processing based on OpenCV [J]. Electronic Testing and Testing,2011,No.229(07):39-41.

[10] Qian Y, Fan Y, Hu W, et al. On the training aspects of deep neural network (DNN) for parametric TTS synthesis[C]. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014: 3829-3833.

[11] Lu Hongtao, Zhang Qinchuan. Application of deep convolutional neural networks in computer vision [J]. Data Acquisition and Processing,2016,31(01):1-17.