Driving Assistance System Based on Deep Learning and Traditional Vision

Zhenwei Bian¹, Tao Yu¹, Xin Zhang¹ and Xiaoyan Gong-Ye¹,*
¹(School of Cyberspace Security, Qufu Normal University Qufu 275165)
*Corresponding Author Email: yigongsd@qfnu.edu.cn

Abstract. Relevant technologies such as computer vision and artificial intelligence are cheaper and easier to implement than detection technologies implemented by hardware such as lidar and radar. Cars are equipped with advanced intelligent driving assistance systems to prevent or reduce traffic accidents. In this context, this paper will identify and analyze the most important traffic lights, vehicles, and lane lines in traffic. Based on ImageNet pre-training, SqueezeNet builds fine-tuned network recognition traffic lights. Aims to achieve an assisted driving system that integrates deep learning and traditional vision. The final model size is only 7.84MB, the recognition accuracy is as high as 94.95%, and the processing speed is 12.4ms / frame. The single-frame processing speed of recognizer of YOLO v3 trained vehicle and classifier of B-CNN trained vehicle is up to 24.47ms. Using computer vision and mathematical operations, image perspective transformation, and polynomial fitting to analyze lane lines has the advantage of reducing cost.

1. Background
As one of the important directions of automobile development, assisted driving has received more and more attention from all walks of life. Related technologies such as machine vision and artificial intelligence are more accurate and cost-effective than the existing detection technologies implemented by LiDAR and radar hardware. They are an important driving force for modern safe transportation. The driving assistance system that integrates deep learning and traditional vision captures video through on-board camera equipment, and analyzes the vehicle condition by means of traffic light detection and analysis, vehicle detection and recognition, and lane line recognition on the captured video. This system is embedded in the driving recorder and then voice Assist the driver to drive, effectively reduce traffic accidents.

2. Traffic Light Detection and Recognition
Algorithms and data scale are factors that affect the speed and accuracy of image processing by machine vision and artificial intelligence. Different algorithms have very different effects when running on different scales of data. When the algorithm is constant, the larger the data size, the slower the processing speed, and the worse the accuracy. The smaller the data size, the faster the processing speed and the higher the accuracy.

2.1. Pretreatment
Generally, due to space constraints, the captured images of on-board equipment placed in the car have a certain pattern, as shown in Figure 1:
According to Figure 1, the image captured by the camera can be divided into three parts, namely the upper (yellow box) traffic light detection area, the middle (red box) vehicle and lane line detection area, and the lower (green box) invalid area (Caused by location influence). In order to avoid missing the detection target, there is an intersection area between the detection boxes. Each module only needs to be identified in the designated area to avoid a large number of invalid calculations, thereby reducing system memory usage and improving calculation efficiency.

2.2. Squeezenet Network Model Based on Imagenet Pre-training

Generally, training a model from scratch requires a large-scale data set and consumes a lot of computing resources. When the target data set and computing power are insufficient, the method of using other trained network model parameters as the initial value of the network to be trained is called pre-training. Research shows [1] that the generalization ability of convolutional neural networks based on ImageNet [2] pre-training will be stronger. Using ImageNet pre-training can accelerate the convergence on the target task, while training the model from scratch requires more iteration cycles to achieve convergence.

SqueezeNet [3] is a lightweight and efficient Convolutional Neural Network (CNN) network model. It can achieve the approximate effect of AlexNet [4], but it reduces the parameters of the AlexNet network by 50 times, which further improves the calculation speed. The deep compression technology is implemented using network pruning, digitization and huffman coding model compression technology. SqueezeNet also combines deep compression [5] technology, which is 510 times smaller than the AlexNet model file, reducing memory requirements.

2.3. Test Analysis

Caffe is a C++/CUDA architecture, an efficient deep learning framework that supports command line, Python and MATLAB interfaces, can seamlessly switch between CPU and GPU, and is good at image processing. The experiment environment installed caffe CPU version and GPU version at the same time, and conducted five different experiments. See the following table 1 Experiment grouping.
Table 1. Experiment grouping

| Label | GPU   | Caffe version | Reading method          | Pretreatment |
|-------|-------|---------------|-------------------------|--------------|
| 1     | No    | GPU           | caffe.io.load_image     | No           |
| 2     | No    | CPU           | caffe.io.load_image     | No           |
| 3     | Yes   | GPU           | caffe.io.load_image     | No           |
| 4     | Yes   | GPU           | opencv                  | No           |
| 5     | Yes   | GPU           | opencv                  | Yes          |

Each group of experiments was run five times to obtain five run times, and the average of the five groups of time was taken as the final result of the experiment. See the following table 2 Data of grouped experiment results.

Table 2. Data of grouped experiment results

| Experiment label | Running speed (ms/frame) | Improved effect |
|------------------|--------------------------|-----------------|
| 1                | 57.0                     |                 |
| 2                | 48.3                     | 15.25%          |
| 3                | 15.9                     | 67.08%          |
| 4                | 14.95                    | 5.97%           |
| 5                | 12.4                     | 17.05%          |

1. Comparing experiment 1 and experiment 2, the Caffe CPU version effectively optimizes the CPU without GPU, and the calculation speed is increased by 15.3%.

2. Comparing Experiment 2 and Experiment 3, it is found that the running speed of caffe with GPU enabled and using the GPU version is 66.9% higher than that of caffe without GPU enabled and using the CPU version.

3. Experiment 3 and Experiment 4 are compared, using the Caffe framework caffe.io.load_image(img_path) function, the speed of reading and processing the image is 0.0159 m/frame, while using the cv2.imread(img_path) function of opencv, the processing speed is optimized by 6%. In-depth research found that caffe.io.load_image(img_path) stores the image as 0-1 float data, the channel order is RGB, cv2.imread(img_path) directly returns the numpy.ndarray object, the channel order is BGR, note that it is BGR, The default range of channel value is 0-255. Due to the different data types and channel order of the two, the bottom layer of Caffe depends on opencv, and many of the encapsulated functions are implemented by opencv functions. When using Caffe, if you use caffe.io.load_image(img_path) to read images, the image format conversion will occur frequently, which increases the amount of calculation and reduces the speed.

4. Comparing experiment 4 and experiment 5, the image is divided into regions using the preprocessing method of view division in Figure 1, and only the traffic light detection area is processed. The processing speed is optimized from 0.01495 m/frame to 0.0124 m/frame, and the performance is improved by 13%.

Based on various optimizations, the experimental computer finally reached 12.4ms/frame processing speed (ubuntu18.04, GTX1060).

In order to clearly figure out the cause of the error, the real results 0 (green light), 1 (red light), 2 (no traffic light) and the predicted results 0 (green light), 1 (red light), 2 (no traffic light) are combined in pairs. There are six kinds of error conditions, namely 0-1 (a green light is judged to be a red light), 0-2 (a green light is judged to be no traffic light), 1-0 (a red light is judged as a green light), 1-2 (a red
light is judged as a no traffic light), 2-0 (no traffic light is judged as a green light), 2-1 (no traffic light is judged as a red light).
The confidence probability chart of the prediction error sample is as follows, see Figure 2.

![Probability vs Type](image)

**Figure 2.** Confidence probability chart of prediction error sample

It can be clearly seen that the error mainly occurs in the confidence probability of the prediction error sample between 0.45 and 0.75, and this interval has the largest error probability. Taking the confidence probability of the prediction error sample as the dependent variable, the number of errors is low when the prediction probability is low and the number of errors is high. When the confidence probability of the prediction error sample is between 0.45 and 0.75, the number of errors is large. The model performs well. Similarly, it is revealed that when the confidence probability is higher than 0.75, misjudgment still occurs, which points out the direction for further optimization.

When training with the same data set, the model of GoogLeNet[6] occupies 41MB, and the model of VGG-16[7] is as high as 528MB. This system chooses ImageNet pre-training, and SqueezeNet builds the model obtained by fine-tuning the architecture. This model only 7.84MB, with 94.955% accuracy, macc (multiply-accumulate) of 7.46G, reaching 12.4ms/frame processing speed (ubuntu18.04, GTX1060), comprehensive consideration of the two important indicators of model size and recognition accuracy Outstanding performance.

### 3. Vehicle Detection and Recognition Module

The target detection and recognition methods in deep learning are divided into two categories: one type of dependent candidate region extraction methods include R-CNN [8](Region-CNN), SPP-NET[9], Fast R-CNN[10], Faster R-CNN[11], R-FCN[12], etc. These algorithms need to first propose the candidate area, and then identify the objects in the candidate area. These algorithms have high accuracy when detecting small targets, but the detection speed is slow; another type of algorithm does not rely on the candidate area but is based on the regression target Detection methods include SSD[13], YOLO[14], etc.

#### 3.1. YOLO v3

YOLO v3 is a fully convolutional network and is the third version of the YOLO algorithm. The huge improvement in algorithm performance is due to the enhancement of the feature extraction network. There are gradient disappearance and gradient explosion problems in deep learning. The deeper the network, the more difficult it is to train successfully.

By drawing on the algorithm ideas of the residual block (Referred to as ResNets) shortcut and the
feature pyramid network [15] (FPN) in the residual network [16], the Darknet19(The backbone network of YOLO v2 to extract features) of YOLO v2 is changed to a deeper Darknet53(The backbone network of YOLO v3 extracting features). The shortcut of the residual network is used extensively. The layer jump phenomenon is more and more common, and the network performance is better than ResNet.

In the prediction stage, the feature pyramid network uses 3 scale feature maps, one down-sampling and two up-sampling, and the cross-entropy loss function is used for category prediction. Without losing speed, the accuracy of target detection is effectively improved. Since it is a multi-label classification, the weighted loss of each label is accumulated as the final loss, where loss = loss_color + loss_direction + 2.0 × loss_type. The vehicle type in the label is the most important. After testing and verification, it is better to set the loss weight of the vehicle type to 2 times.

The cross entropy loss function is shown in formula (1):

$$L = -\sum_{i=1}^{N} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log (1 - \hat{y}^{(i)})$$  (1)

Among them, $y^{(i)}$: the true value of the i-th sample, $\hat{y}^{(i)}$: the true value of y for the i-th sample, and N is the number of samples.

Comparing YOLO v3 with other target detection frameworks, it can be seen that it has excellent performance in detection speed and recognition accuracy[17].

3.2 B-CNN
With the continuous improvement of people's requirements for image classification, fine-grained classification tasks have emerged. Similarly, generally speaking, the deeper the convolutional network, the better the features. However, when the depth reaches a certain level, it becomes more and more difficult to improve the accuracy through depth. B-CNN [18] is a bilinear convolutional network that solves these two problems at the same time. It only needs to get the label of the training sample, and does not need the bounding box of the target and the label of the component, and then the classifier model can be trained. At the same time, it has a higher accuracy rate. The functions of the two convolutional neural networks are equivalent to region detection and feature extraction. On the one hand, B-CNN avoids a large number of manual marking operations in traditional methods, and at the same time obtains a higher accuracy rate.

3.3 Work Process

![Figure 3. Work flow chart of vehicle detection and recognition module](image)

First, initialize the vehicle detection and vehicle recognition model, and preprocess the input
pictures to unify the format. Then detect and identify the picture frame by frame. The model trained by YOLO v3 is used for vehicle detection, and the vehicle is found to generate ROI. ROI filters out prediction frames with higher confidence scores through non-maximum suppression algorithms. Coordinate system conversion removes duplicate targets, and returns the ROI of the maximum score in different target prediction frames as the result. The result is sent to the vehicle classification model trained by B-CNN for classification. In order to balance inference speed and accuracy, the B-CNN basic network uses ResNet-18. The final single frame processing speed theoretically reaches 24.47ms. The working process is shown in Figure 3 working flow chart of the vehicle detection and recognition module. The effect display is shown in Figure 4, the effect diagram of the vehicle detection and recognition module. The final B-CNN model can identify passengerCar, saloonCar, shopTruck, suv, trailer, truck, van, and waggon. When large vehicles are detected in front, the driver will be prompted to pay attention. avoid.

Figure 4. Effect diagram of vehicle detection and recognition module

4. Lane Line Detection and Recognition
In the lane line detection and recognition module, the lane line pixel position is obtained through operations such as image correction, image binarization, and perspective transformation. Then polynomial fitting is performed to obtain the lane line position, the center position of the vehicle-to-vehicle lane line, and the lane curvature. Finally, the voice prompts the driver to avoid pressing the lane line, slowing down and avoiding sharp turns, improving driving safety.

4.1 Image Regularization And Image Correction
The input image format is normalized (the original image size is normalized to the training data image size), considering that the camera is in the car, and the front windshield is not a flat mirror, image correction is required. Use the method provided by opencv to calculate the camera calibration matrix and distortion coefficients through the checkerboard image group, and then use the cv2.undistort() method to get the corrected image.

4.2 Perspective Transformation
The lane line detection area of the corrected image is taken as an ROI, and the ROI is converted into a top view using perspective transformation.
4.3 Binary Image
For the bipolarization of the top view image, by calculating the gradient derivative of the color change in the x-axis direction or the y-axis direction, threshold filtering is performed (above the threshold value is considered as a lane line), and then the binarized map of the lane line is extracted. There are still many noise pixels in the binarized image. A plane coordinate system is established with the image length and width pixels x and y axis. When x is determined, the two pixels with the highest y peak are very likely to be the lane line baseline.

4.4 Quadratic Polynomial To Fit Lane Lines
After obtaining the lane line baseline, the two lane line baselines are respectively located by the sliding window-based target detection algorithm, and the second-order polynomial is used to fit the lane line to obtain the lane line position.

Polynomial fitting formula(2):
\[
\begin{cases}
    f_{\text{right}}(y) = A_{\text{right}}y^2 + B_{\text{right}}y + C_{\text{right}} \\
    f_{\text{left}}(y) = A_{\text{left}}y^2 + B_{\text{left}}y + C_{\text{left}}
\end{cases}
\] (2)

Among them, \(y\) is the ordinate value in the coordinate system, \(f_{\text{right}}(y)\) is the linear equation of the red lane line on the left, \(f_{\text{left}}(y)\) is the linear equation of the blue lane line on the right, and \(A_{\text{right}}, B_{\text{right}}, C_{\text{right}},\) etc. are linear regressions unknown to be sought.

4.5 Calculation of the Distance Between the Vehicle and the Center of the Lane Line
According to the lane line, the center position of the lane line can be obtained. Since the initial camera is placed in the center of the car, the center position of the car can be calculated. The midpoint of the abscissa of the left lane line and the abscissa of the right lane line is the midpoint of the lane line. The distance between the vehicle and the center of the lane line.

4.6 Lane Curvature Calculation
The radius of curvature R of a point on the curve is the radius of the close circle of that point.

The formula for radius of curvature (R):

Suppose that the curvature is \(k\), and the curve descending length \(s\) changes with the angle \(\alpha\). The relationship between \(k\) and \(s\) and \(\alpha\) is \(k = \lim_{\alpha \to 0} \frac{\Delta s}{\Delta x}\), and the specific content is shown in formula (3):

\[
R = \frac{1}{k} = \frac{1 + \left(\frac{dy}{dx}\right)^2}{\frac{d^2y}{dx^2}} = \frac{[1 + (f'')^2]^{\frac{3}{2}}}{f''} \tag{3}
\]

Let the lane line equation be a quadratic fitting curve:

\[
f(y) = Ay^2 + By + C \tag{4}
\]

Among them, \(y\) is the ordinate value of the lane line, and \(A, B,\) and \(C\) are the quadratic coefficient, the first coefficient and the constant term respectively.

From formula (4), \(f', f''\) can be obtained:

\[
\frac{dx}{dy} = f' = 2Ay + B \tag{5}
\]

\[
\frac{d^2x}{dy^2} = f'' = 2A \tag{6}
\]

Then at \(y\), substitute formula (5) and formula (6) into formula (3) to calculate the radius of curvature of the fitted lane line, the radius of curvature is:
When the radius of curvature is less than 500, it is considered to be a larger turn, and the driver is now voiced to prompt the driver to slow down and turn. Lane line detection and recognition relies on computer vision and mathematical operations. Compared with the detection technology implemented by LiDAR and radar, the cost is low and easy to implement. However, the input harsh environment image has poor results and cannot handle harsh road sections and complex road sections. There is still much room for improvement in this module.

5. Hardware System Architecture and Outlook

NVIDIA launched Jetson Nano, an artificial intelligence computer, which looks small and exquisite like a Raspberry Pi, but has very powerful performance. It can provide up to 472 GFLOPS of floating-point computing power and consumes only 5W of power.

Driving assisted systems that integrate deep learning and traditional vision require a lot of parallel computing, and most on-board embedded devices on the market do not have such hardware conditions, which makes the system unable to run or run too slow. Through the hardware combination with Jetson Nano as the computing core, hardware products can be built to get rid of third-party dependence and maximize system performance. See Figure 5 Hardware combination.

![Figure 5. Hardware combination](image)

In order to reduce the cost of hardware equipment, you can make full use of the visual delay, no longer processing each frame, processing across frames. When the system is actually running, there is no need to write video, only voice prompts based on the conclusions drawn, which also reduces the pressure on system hardware performance.
6. Summary
The article mainly introduces the principle and work flow of the assisted driving system that integrates deep learning and traditional vision. The system fully analyzes and utilizes the differences in the application scenarios of different modules, and adopts different optimized processing methods for different modules, mainly including the following aspects:

1. The input image is divided into three parts, namely the traffic light detection area, the vehicle and lane line detection area, and the invalid area (caused by the camera position). Each module only needs to detect the target area, avoiding a lot of useless calculations.

2. Based on ImageNet pre-training and SqueezeNet's architecture fine-tuning, the model obtained by SqueezeNet performs well in the comprehensive consideration of the two important indicators of model size and recognition accuracy. The size of the model is only 7.84MB, and the recognition accuracy is as high as 94.95%, reaching 12.4ms/frame processing speed.

3. The vehicle detection trained by YOLO v3 and the vehicle classification model trained by B-CNN have excellent performance in target detection and fine-grained recognition respectively, meeting the requirements of application recognition accuracy and speed, and the single frame processing speed is as high as 24.47ms.

4. Lane line detection and recognition rely on computer vision and mathematical operations. Compared with the detection technology implemented by LiDAR and radar, the cost is low and easy to implement, and the cost is low.

7. References
[1] Chatfield K, Simonyan K, Vedaldi A, et al. Return of the devil in the details: Delving deep into convolutional nets[J]. arXiv preprint arXiv:1405.3531, 2014.
[2] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.
[3] Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size[J]. arXiv preprint arXiv:1602.07360, 2016.
[4] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.
[5] S. Han, H. Mao, and W. Dally. Deep compression: Compressing DNNs with pruning, trained quantization and huffman coding. arxiv:1510.00149v3, 2015a.
[6] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition, 2015: 1-9.
[7] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
[8] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 580-587.
[9] 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.
[10] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.
[11] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]//Advances in neural information processing systems. 2015: 91-99.
[12] Dai J, Li Y, He K, et al. R-fcn: Object detection via region-based fully convolutional networks[C]//Advances in neural information processing systems. 2016: 379-387.
[13] Warburton D E R, Nicol C W, Bredin S S D. Health benefits of physical activity: the evidence[J]. Cmaj, 2006, 174(6): 801-809.
[14] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.
[15] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object
[10]  \hspace{0.5cm} \text{detection}[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.

[16]  \hspace{0.5cm} \text{He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J]. 2015:770-778.}

[17]  \hspace{0.5cm} \text{WEI Wei, PU Wei, LIU Yi. Application of Improved YOLOv3 in Aerial Target Detection. CEA, 2020, 56(7): 17-23.}

[18]  \hspace{0.5cm} \text{Lin T Y, RoyChowdhury A, Maji S. Bilinear cnn models for fine-grained visual recognition[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1449-1457.}