Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 6, No. 3, 2023

A License Plate Detection and Recognition Method Based on Improved Yolov5

Weiping Yang *, Chao Huang

School of Cybersecurity, Chengdu University of Information Technology, Chengdu, Sichuan, 610200, China
*** Corresponding author:** Weiping Yang (Email: yangflat1997@outlook.com)

Abstract: This study proposes a license plate detection and recognition method based on yolov5. In order to accurately detect the license plate and accurately identify the license plate number, this study adds key point detection based on the yolov5 algorithm to correct the license plate image into a rectangle through perspective transformation, which greatly improves the accuracy of license plate number recognition. Due to embedded deployment, this study simplifies the yolov5 network structure to a certain extent. While reducing the network depth and width, ShuffleNet Block is used to replace the C3 module in yolov5. In the end, this study achieved a mean average precision (mAP) of 89.7% while controlling the network calculation amount to 5GFLOPs. Therefore, this model has broad application prospects in the transportation field.

Keywords: License Plate Detection; Yolov5; Key Point Detection; ShuffleNet.

1. Introduction

License plate detection is an important research direction in the field of computer vision, aiming to achieve automatic detection and recognition of license plates. License plates are an important indicator of vehicle identity and are widely used in traffic management, smart security and other fields. Therefore, license plate detection technology is of great significance in improving traffic management efficiency and ensuring traffic safety. [1-2]

Traditional license plate detection methods are usually based on image processing technologies, such as edge detection, morphological processing, etc. However, these methods are often interfered by lighting conditions, license plate contamination and other factors, making it difficult to achieve accurate license plate detection. With the development of deep learning technology, license plate detection algorithms based on deep learning have gradually become a research hotspot. Deep learning technology can automatically extract image features and has strong robustness, which can effectively solve the problems faced by traditional methods.

Currently, there are two deep learning algorithms for detecting license plates. One is a two-stage detection method represented by R-CNN, SPP-net, Fast R-CNN and Faster R-CNN [3–6]. These methods first generate a series of candidate regions. Classification and bounding box regression are then performed for each region. Another one-stage detection method includes the yolo series algorithm and the SSD [7] algorithm. This detection method turns the target detection task into a single regression problem and directly predicts the location and category of the target on the image. The two-stage target detection algorithm based on candidate region extraction has high accuracy in target detection and can achieve accurate detection of small targets and overlapping targets. However, its biggest disadvantage is that the detection efficiency is low, the generation of candidate regions causes the network feature extraction speed to slow down, and the training and detection time costs are too high to meet the requirements of real-time detection. The single-stage detection algorithm has a relatively simple model structure, can perform real-time detection, and ensures a certain detection accuracy. However, the existing yolov5 algorithm has a certain depth in its model and is difficult to complete feature extraction of small targets. Therefore, it is difficult to extract features of small targets, the detection effect is not good, so the yolov5 algorithm needs to be optimized to some extent.

2. Improved Yolov5 Algorithm

The yolov5 algorithm is divided into four variants from front to deep: yolov5s, yolov5m, yolov5l, yolov5x. Therefore, considering the detection accuracy and real-time detection requirements, the yolov5s algorithm was selected as the basic framework of this research. The basic network structure diagram of yolov5s algorithm is as follows:

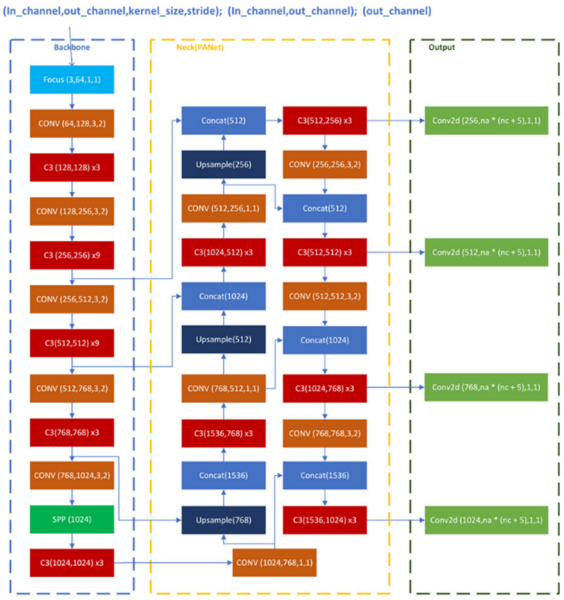

Figure 1. Yolov5 Basic Network Structure

2.1. Lightweight Network

The basic convolution module in Yolov5 is C3 Block, which consists of three Conv modules and a Bottleneck module. Among them, the Conv modules are all 1*1

convolutions, which play the role of dimensionality enhancement or dimensionality reduction. The Bottleneck module consists of a 1*1 convolution and a 3*3 convolution. The first convolution reduces the channels to half of the original size, and the second convolution doubles the number of channels. After the feature map enters C3, it will be divided into two paths. One path passes through Conv and a Bottleneck, and the other path passes through a Conv. Finally, the two paths are processed by concatenating and then input into the Conv module.

However, for the lightweight needs of mobile terminals, the C3 module is too bloated. In order to achieve lightweight networks, more lightweight convolutional networks have been proposed, such as MobileNet [8], ShuffleNet, GhostNet [9], etc. These networks introduce some new network design ideas, which can maintain the accuracy of model detection to the greatest extent and effectively reduce the parameter amount and computational complexity of the model to a certain extent, which is of great significance for the deployment of mobile terminals.

Among the existing lightweight networks, the MobileNet series network and the ShuffleNet series network, as the most representative networks, have a significant effect on reducing the complexity of the network structure. MobileNet is a lightweight convolutional neural network proposed by Google. Its core goal is to reduce the size and computational complexity of the model as much as possible while maintaining model accuracy. To achieve this goal, MobileNet uses depthwise separable convolutional layers, which consist of DW convolutional layers and pointwise convolutional layers. The design idea of DW separable convolution is to reduce the amount of calculation and model size. ShuffleNet is a lightweight convolutional neural network proposed by Stanford University. It uses a technique called "channel rearrangement" and a special group convolution operation called "channel-wise group convolution". This operation can reduce the amount of calculation while maintaining the relationship between each channel. Different from MobileNet, ShuffleNet adopts different design ideas and aims to reduce the amount of calculation and model size.

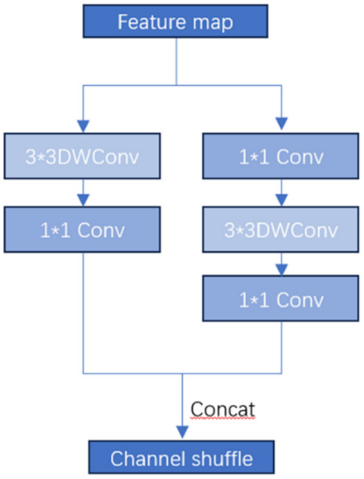

Figure 2. ShufleNet Block

In this study, we chose to use ShuffleNetV2 [10] as an alternative to C3 Block. It not only considers the impact of computational complexity on inference speed, but also takes into account memory access costs and platform characteristics. The figure below shows the basic structure of the ShuffleNet Block used in this study. When the feature map is input, it is divided into two branches. One branch goes through a depth convolution and a 1*1 convolution, and the other goes through a 1*1 convolution in turn. Convolution, a 3*3 DW convolution and a 1*1 convolution, and finally the feature maps of the two branches are concatenated and input into the channel rearrangement structure.

2.2. Activation Function

In the backbone of yolov5, we introduced a new nonlinear activation function Hardswish. It is an activation function that is similar to the Swish activation function and has similar performance to Swish but requires less calculation. Compared with other activation functions, such as ReLU, Sigmoid, etc., Hardswish has significant improvements in speed, accuracy and robustness.

Due to its excellent performance, Hardswish is widely used in deep learning models, especially in scenarios that require processing large amounts of data and computing resources. For example, in the fields of image classification, target detection, speech recognition and other fields, using Hardswish can effectively improve the accuracy and running speed of the model. The formula is as follows:

$$ReLU6(x) = min(max(0, x),6) \tag{1}$$

$$hardswish(x) = x \times \frac{ReLU6(x+3)}{6} \tag{2}$$

Compared with the SiLU() activation function used in the original yoloV5, hardswish greatly reduces computational complexity, avoids power exponential operations, improves inference speed, and has less impact on model detection accuracy.

2.3. Key point Detection

Key point detection is a computer vision technology whose purpose is to find meaningful feature points in images or videos and locate them at each feature point. These feature points are often called key points because they are unique in the image and can be used to represent the content or shape of the image. Key point detection has applications in many fields, including face recognition, gesture recognition, robot navigation, image stitching, and video analysis. In the license plate detection, the four corner points of the license plate are added as key points, and perspective transformation is used to restore the visually deformed quadrilateral license plate image captured by the camera into a rectangle, which is more conducive to accurate recognition of the license plate number. Perspective transformation refers to the use of the collinear condition of the perspective center, image point, and target point to rotate the image-bearing surface (perspective surface) at a certain angle around the trace (perspective axis) according to the perspective rotation law, destroying the original projection light The wire harness can still maintain the same transformation of the projected geometry on the shadow surface. The picture below shows the renderings before and after perspective transformation.

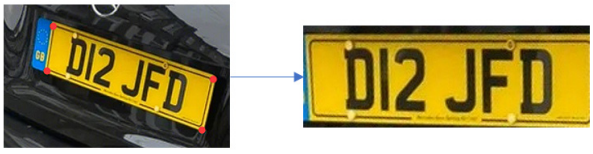

Figure 3. Perspective Transformation

By converting the irregular quadrilateral license plate image into a rectangular license plate image, the deformed license plate number on the license plate will also be corrected, which is more conducive to improving the accuracy of license plate number recognition.

2.4. License Plate Number Recognition

After detecting the license plate, the license plate image needs to be input into the OCR model to identify the license plate number. In this part, we selected the CRNN [11] (Convolutional Recurrent Neural Network) algorithm. CRNN is a convolutional recurrent neural network structure used to solve problems based on Image sequence recognition problem, especially scene text recognition problem. CRNN is mainly used to recognize text sequences of variable length end-to-end. It does not need to cut individual words first, but transforms text recognition into a time-series-dependent sequence learning problem, that is, image-based sequence recognition. The structure of CRNN mainly consists of convolutional layer, recurrent layer and transcription layer. First, the feature sequence is extracted from the input image through the convolutional layer; then, the recurrent layer predicts the label distribution of the data set; finally, the transcription layer transcribes the prediction results of each picture of the recurrent layer into the final label sequence. During the training process, CRNN adopts the connectionist temporal classification (CTC) [12] loss function to achieve sequence learning without alignment. CRNN performs well in text recognition tasks, with high recognition accuracy and strong robustness.

3. Performance Evaluation

3.1. Dataset

The data set used in this study is the CCPD data set. The CCPD data set is a benchmark data set for license plate recognition. It contains various license plate images from mainland China, including ordinary license plates, new energy license plates, armed police license plates, embassy license plates, etc. The purpose of this dataset is to provide a standard testing platform so that researchers can compare and evaluate different license plate recognition algorithms on the same dataset. In this experiment, all images in the data set were divided into training sets and validation sets at a ratio of 9:1.

3.2. Experiment Environment

The experiment runs on the Ubuntu20.04LTS operating system, using the PyTorch deep learning library, and the python version is 3.8. In terms of hardware, the CPU model is AMD Ryzen 7 5700X, the GPU is NVIDIA GeForce RTX3060Ti 8GB, and the memory is 32GB.

3.3. Indicators for Model Evaluation

Precision (P), Recall (R), Average Precision (AP) and mean Average Precision (mAP) are the main evaluation indicators of this experiment. Their calculation formulas are as follows

$$P = \frac{TP}{TP+FP} \tag{3}$$

$$R = \frac{TP}{TP+FN} \tag{4}$$

$$AP = \int_0^1 p(r) \tag{5}$$

$$mAP = \frac{1}{n}\sum_{i=1}^{n} APi \tag{6}$$

P calculates the proportion of correct positive samples to the total samples and is used to evaluate the accuracy of model detection. R calculates the proportion of detected positive samples to total positive samples and is used to evaluate the detection rate of the model. AP is used to average the Precision under different confidence thresholds. mAP is the AP calculated based on multiple preset thresholds, and finally all APs are averaged. In this paper, we set the IOU threshold to 0.4 and the confidence threshold to 0.25.

In addition to accuracy, computational complexity is also an important evaluation index of this model. Only with sufficient speed can real-time detection be achieved. In order to measure the complexity of the model, we will calculate the detection time of each frame in ms.

3.4. Experimental Results

The model input size used in the original yolov5s training is 640*640, the batch-size is 32, the training Epoch is 200, and the learning rate is 0.001. The final training P, R, AP, mAP, and average inference time are 97.3%, 94.2%, 94.1%, 92.1%, 51ms. The improved Yolov5 in this article uses the same training hyperparameters. The final P, R, AP, mAP and average inference time obtained by training are 96.1%, 92.1%, 91.5%, 89.7%, 25ms. The final indicators of the two models are as follows:

Table 1. Experimental Results

Model_name	P	R	AP	mAP	time
Yolov5s	97.3	94.2	94.1	92.1	51
Improved Yolov5s	96.1	92.1	91.5	89.7	30

Figure 4. license plate detection

4. Conclusion

In license plate detection and recognition, due to the limitations of the working environment, mobile embedded devices are more suitable for detection tasks. However, embedding equipment has limited computational examples. Therefore, in order to balance detection accuracy and model inference speed, this paper proposes a license plate detection and recognition model based on improved yolov5, making certain improvements in the model backbone network, model structure, and activation function. , and key point detection is added to the detection, which greatly improves the model inference speed while ensuring the accuracy of model detection and license plate number recognition. In the CCPD data set, compared with the original yolov5s, the number of model parameters in this article has dropped by 50.9%, the inference time has dropped by 26ms, and when precision is used as the standard, the detection accuracy has only dropped by 1%. Through experiments, we found that the model in this study is more suitable for license plate detection and

recognition tasks, achieving a good balance in terms of accuracy and speed.

In the future, we will further optimize the model and explore optimization methods to improve its detection accuracy without increasing the model parameters, so that the network can be more competent in license plate detection tasks.

References

[1] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp.779–788.

[2] Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices [C]// Proceedings of the IEEE conference on computer vision and pattern recognition. 2018:6848-6856.

[3] He K, Gkioxari G, Dollár P, et al. Mask r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2961-2969.

[4] Girshick R. Fast r-cnn[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1440-1448.

[5] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. Advances in neural information processing systems, 2015, 28.

[6] 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.

[7] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector [C]// Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer International Publishing, 2016: 21-37.

[8] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications [J]. arXiv preprint arXiv:1704.04861, 2017.

[9] Han K, Wang Y, Tian Q, et al. Ghostnet: More features from cheap operations [C]// Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 1580-1589.

[10] Ma N, Zhang X, Zheng H T, et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design[C]// Proceedings of the European conference on computer vision (ECCV). 2018: 116-131.

[11] Shi B, Bai X, Yao C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 39(11): 2298-2304.

[12] Graves A, Fernández S, Gomez F, et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks[C]//Proceedings of the 23rd international conference on Machine learning. 2006: 369-376.