Research On the Application of Deep Learning-Based License Plate Computer Recognition System in Traffic Scenarios

Xingxing Li*, Chao Duan, Panpan Yin, NingXing Wang
School of Engineering, Guangzhou College of Technology and Business, FoShan, 528138, China.

*Corresponding author: wslxx@jxstnu.edu.cn

Abstract. As the development of machine vision technology matures, this paper builds a set of license plate recognition algorithms based on deep learning algorithms and applies them in traffic monitoring scenarios, which can accurately identify the license plate information of passing vehicles at traffic intersections, and the detection accuracy mAP can reach more than 99%. The recognition accuracy can reach more than 94%, the detection algorithm uses YOLOV4, which can accurately detect the position information in the image where the license plate is located, the recognition algorithm uses CRNN, the detected ROI area is cut out for text recognition, and the algorithm is deployed and used Deployed with the tensorflow acceleration library and deployed end-to-end on the NVIDIA Jetson AGX Xavier embedded edge computing device. It only takes 17ms to identify all the license plate number information in an image.

Keywords: Deep learning algorithms, traffic monitoring scenarios, license plate recognition, CRNN.

1. Introduction
License plate recognition system is widely used in life, especially in parking lot, toll station and other occasions. With the implementation of intelligent traffic, license plate recognition can also be applied to the monitoring equipment of traffic road intersection. Real-time recognition of the license plate information of the past vehicles can be more convenient for the coordinated control of traffic vehicle information. There are many challenges to license plate recognition, such as blurred image, poor light conditions, changes of license plate numbers (such as special characters in license plates in China and Japan), license plate deformation, weather impact (such as rainy and snow weather), and the number of characters on the license plate.

In previous OCR tasks, the recognition process is divided into two steps: word segmentation and classification. Generally speaking, a series of text files are cut into a single font by projection method and sent to CNN for text classification. However, this method is a bit out of date, and the more popular one is the end-to-end character recognition based on deep learning, that is, we do not need to explicitly add the link of character cutting, but turn the character recognition into the problem of sequence learning. Although the input image scale is different, the text length is different, but after DCNN and...
RNN, after a certain translation in the output stage, in other words, text cutting is also integrated into deep learning [1].

Nowadays, there are two main technologies of end-to-end OCR based on deep learning: crnn OCR and attention OCR. In fact, the main difference between the two methods lies in the final output layer (translation layer), that is, how to transform the sequence feature information learned by the network into the final recognition result. In the feature learning stage, these two mainstream technologies adopt the network structure of CNN + RNN. The alignment method of crnn OCR is CTC algorithm, while the alignment method of attention OCR is attention mechanism. This paper will introduce the more widely used crnn algorithm [2].

In this paper, yolov4[3] target detection algorithm and crnn text recognition are used to detect the license plate of a complete vehicle area by using yolov4, and then the ROI area where the license plate is located is detected. Then, the crnn text recognition algorithm is used to identify the license plate content in this ROI area. The end-to-end flow can be realized by using this method of detection and text recognition, and can ensure the real-time deployment.

2. Principle
In this paper, we use the yolov4 [3] target detection network to locate the license plate in the complete vehicle area. The yolov4 target detection is improved on the previous version of Yolo.

Bag of freebies (BOF) for backbone networks: Cutmix and Mosaic data enhancement, DropBlock regularization and class label smoothing; Bag of special (BOS): Mish activation, CSP and multiinput weighted residual connection (MiWRC) for backbone networks; Bag of freebies (BOF) for detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data enhancement, self-confrontation training, eliminate grid sensitivity, multiple anchors, cosine annealing scheduler for a true value, optimization of superparameters and random training shape; Bag of special (BOS) for detectors: Mish activation, SPP block, SAM block, PAN path aggregation block, and DIoU NMS.

2.1. CRNN network structure
The structure of crnn character recognition network consists of three parts, as shown in Figure 1.

![Figure 1. Schematic diagram of CRNN network structure.](image-url)
1. Convolution layer, using CNN, is used to extract feature sequence from input image; 2. Loop layer, using RNN, is used to predict the label (real value) distribution of feature sequences obtained from convolution layer; 3. In the transcription layer, CTC is used to transform the label distribution obtained from the circulation layer into the final recognition result through de integration and other operations;

CRNN uses the modeling method of LSTM + CTC in speech recognition for reference. The difference is that the features input into LSTM are replaced by the image feature vectors extracted from CNN network from the acoustic features (MFCC, etc.) in speech field. The greatest contribution of CRNN algorithm is to combine the potential of CNN for image feature engineering with the potential of LSTM for serial recognition. The RNN training algorithm based on connectionist temporal classification (CTC) is superior to the traditional speech recognition algorithm in the field of speech recognition. Some scholars try to apply CTC loss function to OCR recognition, and CRNN is one of the representative algorithms. CRNN algorithm inputs 100 $\times$ 32 normalized height entry images, extracts feature map based on 7-layer CNN (vgg16 is commonly used), divides the feature map into map to sequence, and inputs 512 dimensional features of each column to bidirectional LSTM of 256 units in two layers for classification. In the training process, under the guidance of CTC loss function, the approximate soft alignment between character position and class mark is realized.

It not only extracts robust features, but also avoids single character segmentation and single character recognition which are very difficult in traditional algorithms through sequence recognition. At the same time, serialization recognition also embeds temporal dependence (using corpus implicitly). In the training phase, CRNN scales the training image by 100 $\times$ 32 $(w \times h)$; In the test phase, CRNN maintains the scale of input image size, but the image height must be unified as 32 pixels. The size of convolution feature map dynamically determines the LSTM sequence length.

CRNN network processing input has an image. In order to input features to recurrent layers, do the following processing, as shown in Figure 2. First, the image is scaled to 32 $\times$ w $\times$ 1. Then it's changed to 1 after CNN $\times$(w/4) $\times$ 512, but for LSTM, set $t = (w / 4), d = 512$ to input the feature into LSTM. Finally, LSTM has 256 hidden nodes, which become t-length after LSTM. After softmax processing, each element of the column vector represents the corresponding character prediction probability. Finally, the prediction result of T is combined into a complete recognition result.

The problem that needs to be solved in CRNN is that the length of image text is variable, so there will be a problem of alignment decoding, so RNN needs an additional partner to solve this problem, which is the famous CTC decoding partner. The architecture of CRNN is CNN + RNN + CTC. CNN extracts image pixel features, RNN extracts image temporal features, and CTC summarizes the connection characteristics between characters [2].

2.2. CTC character decoding
When RNN is classified in sequence, there will inevitably be a lot of redundant information, such as a letter is identified twice continuously. This requires a set of redundancy mechanism. However, it is also difficult to see the method of redundancy for two consecutive letters. For example, the words like cook and geek have a blank mechanism to solve this problem. Here is an example.
As shown in Figure 3 above, we need to recognize the handwritten image with the label "ab". After CNN + RNN learning, the length of the output sequence vector is 5, that is, t0 ~ t4. At this time, we need to translate the sequence into the final recognition result. The first problem we encounter in translation is how to translate five sequences into two corresponding letters? How to solve the repeated sequence? How to map the sequence of spaces between words? These are the problems that CTC needs to solve.

From the naked eye, we can see that t0, t1, t2 should be mapped to "a", t3, t4 should be mapped to "b". If it combines the repeated characters into one output, "aaabb" will be combined into "ab" output. However, there are some problems in the merging mechanism of such sub images. For example, when our label image is "aab", our sequence output may be "aaaaaaabbb", so we can't determine whether the text should be recognized as "aab" or "ab".

In order to solve this ambiguity, CTC puts forward the mechanism of inserting blank. For example, if we use the symbol of "-" to represent blank, then if the label is "aaa-aaaabb", it will be mapped to "aab", and "aaaaa-abb" will be mapped to "ab". By introducing the blank mechanism, we can deal with the problem of repeated characters.

But we also note that "aaa-aaaabb" can be mapped to "aab". Similarly, "aa-aaaabb" can also be mapped to "aab", that is to say, there are many different character combinations that can be mapped to "aab". To sum up, a label has one or more paths. For example, in the following "state" example, there are several different paths mapped to "state":

\[
B(\pi_1) = B(\neg
\neg
stta - t - - e) = state
\]
\[
B(\pi_2) = B(sst - a aa - tee) = state
\]
\[
B(\pi_3) = B(sttaa - tee) = state
\]
\[
B(\pi_4) = B(sst - aa - t - - - e) = state
\]

As mentioned above, the RNN layer outputs the probability matrix in the sequence, so for the output path \( \pi \) the probability of is the product of the probabilities of each sequence. So, to get a label, we can have multiple paths to get it. Intuitively, we need to maximize the probability of outputting a text image to the network. Because paths are mutually exclusive, the conditional probability of a label sequence is the sum of the probabilities of all paths mapped to it, as shown in Formula 1.

\[
p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x)
\]  

Among \( \pi \in B^{-1}(l) \), all paths can be merged into \( l \). By mapping the sum of the probabilities of \( B \) and all candidate paths, CTC does not need to segment the original input sequence accurately, which makes it possible for RNN layer to translate tasks with the output sequence length > label length. CTC
can be associated with any RNN model, but considering that the labeling probability is related to the whole input string, rather than only to the fragments in the front small window, the bidirectional RNN / LSTM model is more suitable. CTC calculates loss to find the most likely character in the pixel region.

3. Experiments
In this paper, we use the CCPD license plate data set. Because the CCPD license plate data set contains a large number of license plates in Anhui Province, in order to train a general license plate recognition model, we use high-resolution simulation images to generate license plates with the same proportion in each province on the CCPD data set, paste them on the original license plate position, and train them. The detection accuracy map on the test set can reach 99%, The recognition accuracy reaches 97%, and the effect on the test set is shown in Figure 4 below.

![Figure 4. License plate recognition effect picture.](image)

In Figure 4, firstly, the specific location of the license plate is detected by yoov4, and then the content of the license plate in the rectangular box is recognized by crnn text recognition network. We can see that the license plate of each province can be recognized. Let the model be applied to the perspective of traffic monitoring, and the effect is shown in Figure 5.

![Figure 5. The effect of vehicle license plate recognition from the perspective of traffic.](image)
4. Conclusion
From the perspective of traffic monitoring, the area of license plate in the image is very small, so we need to use the detection model to detect the vehicle position first, and then use the license plate detection and recognition model in the vehicle area to recognize the license plate content. In the actual project, only the model detection can recognize the license plate near, and the license plate far away is too small to be distinguished by naked eyes. From the picture, we can see that the near license plate can be accurately identified.

Acknowledgments
This work was financially supported by fund project, that is, Young Talents in Higher Education of Guangdong, China, (No. 2019KQNCX231 and No. 2019KQNCX232).

References
[1] Zherzdev S, Gruzdev A. LPRNet: License Plate Recognition via Deep Neural Networks. 2018.
[2] Shi B, Xiang B, Cong Y. 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 & Machine Intelligence, 2016, 39(11):2298-2304.
[3] Bochkovskiy A, Wang C Y, Liao H. YOLOv4: Optimal Speed and Accuracy of Object Detection[J]. 2020.