A CNN-Based Approach for Automatic License Plate Recognition in the Wild

Meng Dong[1]
dongmeng@mail.ustc.edu.cn
Dongliang He[1]
hedl@mail.ustc.edu.cn
Chong Luo[2]
cluo@microsoft.com
Dong Liu[1]
dongeliu@ustc.edu.cn
Wenjun Zeng[2]
wezeng@microsoft.com

[1] Department of Electronic Engineering and Information Science
University of Science and Technology of China
Hefei, Anhui, P.R. China
[2] Microsoft Research Asia
Beijing, P.R. China

Abstract

In this paper, we address automatic license plate recognition (ALPR) in the wild. Such an ALPR system takes an arbitrary image as input and outputs the recognized license plate numbers. In the detection stage, we adopt a cascade structure comprising of a fast region proposal network and a R-CNN network. The R-CNN network not only eliminates false alarms but also regresses corner positions for each detected plate. This allows us to estimate an affine transformation matrix to rectify the extracted plates. In the recognition stage, we propose an innovative structure composed of parallel spatial transform networks and shared-weight recognizers. The system is trained and evaluated on a Chinese license plate dataset with over 18K images. Results show that our detector performs better than faster R-CNN (VGG) which is 1.5x slower in testing and 57x larger in model size. The recognizer is also significantly better than existing solutions, reducing 57.5% of the errors of a state-of-the-art character sequence encoding scheme.

1 Introduction

Automatic license plate recognition (ALPR) is an important computer vision task. It finds many valuable applications in smart cities, including electronic payment, on-road law enforcement and surveillance. Conventionally, an ALPR system is composed of four stages [4], namely image acquisition, license plate extraction, license plate segmentation and character recognition. When the image acquisition stage is under control, the entire task is pretty much solved. Basic algorithms relying on handcrafted image features could work pretty well[1, 4]. However, if the input image is taken in the wild, ALPR remains to be a challenging problem because license plates could appear at any places of an image with complicated backgrounds. Besides, the plates could be small, tilted, blurred or under uneven illumination. Fig.1 shows a few examples of the input images and zoomed-in license plates we handle in this work. The uncontrolled image acquisition stage brings great difficulties to the subsequent stages of ALPR.

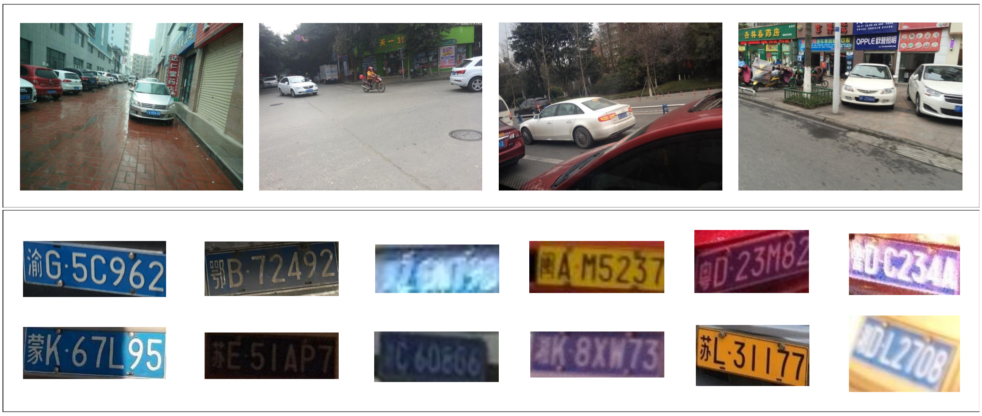

Figure 1: Examples of the input images and zoomed-in license plates we handle in this work.

License plate extraction can be solved by text localization and classification in cascade. Recently, with the explosive interests in deep learning, there have emerged many CNN-based text localization solutions[8, 9, 10, 22, 23]. Using such text spotting networks for license plate detection is a feasible solution, but not an efficient one for our task. This is because images taken in the wild exhibit complicated background with various types of text blocks, and license plates only occupy a very small portion of them. We believe that a dedicated license plate detector is more efficient.

After the license plate is extracted, recognition can be achieved by character segmentation and recognition [4]. Accurate character segmentation is of paramount importance to the subsequent recognition stage, but conventional methods based on handcrafted features only work well when the license plate is accurately extracted and well rectified. Again, this is difficult for images taken in the wild. Although research efforts have been made on recognizing text in natural images[12, 14, 18], these methods mainly focus on handwriting or printed words. The input of these methods is assumed to be well-trimmed image patches, and few of them take the localization error into consideration. In literature, the system designed by Li and Shen [15] appears to be the most related work to ours. After license plate detection, it treats license recognition as a sequence labeling problem and solves it by leveraging CNN and bidirectional long short term memory (LSTM) network. However, we discover that this method is very fragile to distortions caused by viewpoint change.

We aim to design a fast and accurate ALPR system for images taken in the wild. Contributions are made in both license plate detection and recognition. In the detection stage, we adopt a cascade structure composed of region proposal network (RPN) and R-CNN classifier. The RPN network takes downsampled images as input, so it is fast and effective. Once the region of interest (RoI) is proposed, the corresponding image patch is cropped from the original high-resolution image. The R-CNN not only performs classification, but also regresses the four corner positions for each positive license plate. The corner positions are used to calculate an affine transformation matrix for license patch rectification. This step will be shown to be very important as it significantly improves the subsequent recognition accuracy. License plate recognition is achieved by a parallel spatial transform network (STN) with shared weight classifiers. The unsupervised STN implicitly performs character segmentation and focuses on the image pixels of each character. Finally, the shared weight classifier is used to recognize each character. The innovative shared-weight recognizer significantly reduces model size, and more importantly it makes better use of the limited training data.

We currently focus on Chinese license plates although the proposed methods can be extended with small modifications to detect and recognize license plates in other countries. We have collected and annotated our own dataset with more than 18K images. Extensive experiments show that our solution outperforms state-of-the-art solutions in both license plate detection and recognition.

The rest of the paper is organized as follows. Section 2 and Section 3 present the design of license plate detector and recognizer, respectively. Section 4 details the training procedure and Section 5 presents the evaluation results. We finally conclude in Section 6.

2 License Plate Detection

License plate detection is treated as an object detection problem. For CNN-based object detection, both proposal-based methods [6, 7, 16, 20] and single shot solutions[5, 17, 19] have shown their respective advantages. In this work, we adopt a proposal-based method, as shown in Fig. 2. The cascade structure is inspired by a recent face detector [2]. First, a light-weight RPN network [20] takes the downsampled image as input and generates license plate candidates. Then, a sampler extracts the regions of interest (RoIs) from the original high-resolution image. The extracted patches are classified by the R-CNN network. For each detected plate, the regression branch predicts the coordinates of four corner points. This information will be very useful in the subsequent recognition stage.

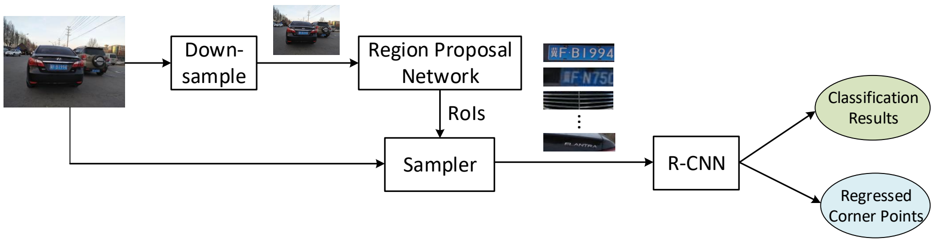

Figure 2: The proposed detector first generates region proposals from the downsampled image, and then samples image patches from the original image according to the RoIs. Last, the R-CNN module classifies each patch and regresses the corner points if the patch is classified as a license plate.

The reasons why we do not use the state-of-the-art faster R-CNN framework [20] are two-fold. First, the features extracted from the RPN is too coarse for precise classification and corner regression. Second, license plates are usually "small" in an image. Doing separate feature extraction will not cost too much. Besides, such a design allows the RPN to take a downsampled image as input, greatly reducing the complexity of RPN.

Specifically, following the design in [2], the base network of RPN consists of three convolution layers followed by inception block 3a and 3b of GoogLeNet v2 [21]. This design strikes a good balance between recall performance and runtime speed. Configuration of the three convolution layers in RPN are summarized in Table 1. RPN aims at generating license plate candidates and regressing their bounding box locations. The loss of RPN network consists of classification loss and smooth-L1 bounding box regression loss as defined in [20]. Note that RPN has a feature stride of 8 and the receptive field is not small. In addition, its input is a downsampled image for efficiency, so its corresponding feature maps of license patch lacks locality.

Table 1: Configuration of First Three Conv Layers of RPN

	conv1	pool1	conv2_1	conv2_2	pool2
Stride	2	2	1	1	2
Kernel size	7×7	2×2	1×1	3×3	2×2
Activation	ReLU	-	ReLU	ReLU	-
# Output channel	64	64	128	288	288

The R-CNN network does not share features with RPN. Instead, it takes as input the RGB values of each license patch, which is sampled from the original image and then scaled to resolution of 48×120. As such, finer local features can be utilized for better classification and corner regression. Feature extraction in the R-CNN network is achieved by three convolution layers. The configuration is summarized in Table 2. The extracted features are fed into two fully connected branches, performing classification and regression, respectively. The loss function of the R-CNN contains two part, one is the cross-entropy classification loss and the other is the smooth-L1 corner point regression loss:

$$L_{R-CNN} = \frac{1}{N}\Sigma_{i=1}^{N} CrossEntropy(p_i, p_i^*) + \lambda\frac{1}{N}\Sigma_{i=1}^{N} p_i^* SmoothL_1(t_i, t_i^*) \tag{1}$$

where N is the number of training samples, p_i^* equals 1 when the i^{th} sample is positive, otherwise 0. $t_i = [t_{i,x1}, ..., t_{i,x4}, t_{i,y1}, ..., t_{i,y4}]$ is the predicted corner shifts and t_i^* is the groundtruth. The x-axis (or y-axis) shift values are normalized by the width(or height) of its corresponding proposal bounding box, for example, when $t_{i,x1}$ is 0.05 and the width of the i^{th} proposal box is 100, the shift value is 5 pixels. In our R-CNN, the groundtruth shift values are allowed to be negative or greater than 1 such that the regressor can imagine where the actual corner points are if the license plate proposal is truncated.

Table 2: Configuration of R-CNN network

	conv_1	pool_1	conv_2	pool_2	conv_3	fc1	fc2
Stride	1	2	1	2	2	-	-
Kernel size	5×5	2×2	5×5	2×2	3×3	-	-
Activation	ReLU	-	ReLU	-	ReLU	ReLU	ReLU
# Output	64	64	64	64	128	64	32

Another thing worthy of mention is the sampler, which crops proposal patches for R-CNN input. Candidate proposals have different aspect ratios and sizes. In order to keep the original aspect ratio of proposals when they are fed into R-CNN, the sampler crops candidate patch with aspect ratio of 1:2.5 centered at the proposal box. Concretely, if aspect ratio of a proposal box is greater(or smaller) than 1:2.5, the horizontal (or vertical) boundaries of the proposal box are symmetrically extended. If boundaries expand out of the image region, we pad with value 128.

3 License Plate Recognition

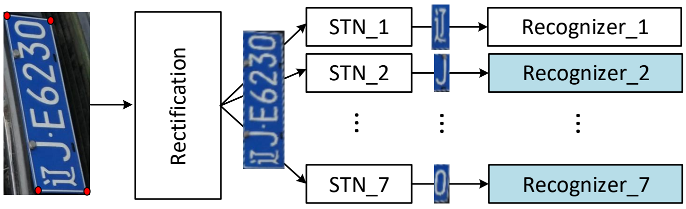

Figure 3: Illustration of the proposed recognizer. The detected plates are first rectified according to the estimated corner points. Then, the image patch for each character is extracted by the STNs and is passed to the recognizers. The cyan branches indicate shared weight recognition networks.

The proposed license plate recognition framework is shown in Fig. 3. The input of this stage is the license plate image patch and its four corner coordinates generated by the previous stage. We first perform license patch registration to obtain a rectified license plate of resolution 48×120. The unsupervised STN [13] implicitly performs character segmentation. Each of them attentions to the pixels of each corresponding character. The output of the parallel STN are segments of the input license patch. Finally, the recognizer recognizes each segment.

For clarity, we illustrate the recognition part using common Chinese license plates. A common Chinese license plate number contains seven characters. It starts with a Chinese character denoting one of the 31 provinces followed by an uppercase letter denoting the city. The remaining five characters are combinations of Arabic numerals and uppercase letters except 'O' and 'I'. After the STN sub-network, we will need seven recognizers. While the first recognizer is separately trained for 31 Chinese characters, the rest six recognizers share weights. This is an important innovation in our design, because weight sharing has at least two advantages. First, it saves model size. Second, it makes full use of the limited training data. Without weight sharing, independent recognizer can only be trained with the digits and letters appear at its corresponding location. It will easily suffer from insufficient training data or unbalanced training sample distribution.

Table 3: Configuration of recognizer base network. BN denotes batch normalization[11]

	conv1	pool1	conv2	pool2	conv3	conv4	pool3	conv5	fc
Stride	1	2	1	2	1	1	2	1	-
Ker. size	5	2	5	2	3	3	2	3	-
Activation	BN	-	BN	-	BN	BN	-	BN	ReLU
# Output	64	64	64	64	128	128	128	256	-

Specifically, the rectification process is performed by an affine transformation:

$$\begin{bmatrix} h^s \\ w^s \end{bmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{bmatrix} h^t \\ w^t \\ 1 \end{bmatrix} \quad (2)$$

where (h,w) denotes the height and width coordinates, superscript s and t denote source patch and target patch, respectively. We have the four corner coordinates of license plate, and our target four corner points are (0,0), (48,0), (0,119) and (48,119). The affine transformation matrix is estimated using the RANSAC algorithm [3]. The parallel STN sub-networks are trained to extract character level segments also by affine transformation defined in Equation 2. In detail, the six parameters of each affine transform matrix for cropping character segments are predicted by network, then a spatial transformer layer takes the six parameters and rectified license patch as input and outputs a 48×24 character segment. The training of STN sub-networks is only supervised by the groundtruth recognition results. In our implementation, the configuration of the seven STNs are summarized in Table 4.

Table 4: Configuration of STN network

	conv1	pool1	conv2	pool2	conv3	fc1	fc2	fc3
Stride	1	2	2	2	2	-	-	-
Kernel size	5×5	2×2	5×5	2×2	3×3	-	-	-
Activation	ReLU	-	ReLU	-	ReLU	ReLU	ReLU	ReLU
# Output	64	64	64	64	128	64	32	6

The seven recognizers share the same network architecture of five convolution layers and a fully connected layer to learn feature representation. The features are then fed into a fully connected layer for classification. Per-character classification cross-entropy loss is adopted in training the recognizer. The details about the feature representation network structure are depicted in Table 3.

Theoretically, with shared weights recognizers, we should multiply the a priori probability to the likelihood estimated by the recognition sub-network. For simplicity, we assume that the Arabic numerals and uppercase letters are uniformly distributed in the 3^{rd} to 7^{th} characters. For the 2^{nd} character, the prior probabilities of Arabic numerals digits are zeros.

4 Training

4.1 Detection Network

Dataset: We collected a dataset of 18699 images taken in various background and with various resolution. Each image has at least one recognizable license plate in its original resolution. We rescale all the images to 960×720 for the convenience of processing. The dataset is divided into a training set ($Train\_Det$) with 12700 images and an evaluation set ($Eval\_Det$) with 5999 images.

Training Strategy: Our detection and recognition networks are trained separately. For the detection network, we first train the RPN with anchor areas of $[600, 1200, 1800, 2800, 8100]$ and anchor aspect ratios of $[1/2, 1/2.7, 1/3.5]$ using $Train\_Det$ for 100K iterations. Learning rate is set to 0.001 and 0.0001 for the first and last 50K iterations, respectively. The mini-batch size is 64 where positive sample ratio is 0.25 and IoU threshold is set to 0.5 to distinguish positive and negative anchors. Training image input size is 360×480. Then, we fix the RPN and train the R-CNN part for 100K iterations with the same mini-batch and learning rate settings as RPN. λ is set as 1 to calculate the loss function of R-CNN.

4.2 Recognition Network

Dataset: To generate dataset for the recognition part, we apply the trained detector on *Eval_Det* and get 5835 detected license patches. We annotate the groundtruth recognition results of 4693 patches (denoted as *Train_Rec*) for training. The rest 1142 patches (denoted as *Eval_Rec*) are left for evaluation. Note that *Train_Det* is only involved in detection and not used for recognition. This is because we need to train the parallel STN to perform segmentation when the input license plates are not perfectly aligned. However, the plates in *Train_Det* usually have very accurate regressed corner points, so they do not help in training the parallel STN. Besides, in order to tackle the data unbalance problem of *Train_Rec*, we add synthetic license patches to it such that the number of license plates of each province in *Train_Rec* is at least 200.
Training Strategy: Using the detected corner points, license patches and annotated recognition results of *Train_Rec* as input, we train the recognition network for 40K iterations. The learning rate is 0.0005 and 0.00005 for the first and last 20K iterations, respectively. Batch size is set to 64.

5 Evaluation

5.1 Detection Results Comparison

Different from the state-of-the-art detector Faster R-CNN [20], our detector does not share CNN feature between RPN and R-CNN. Besides, we use a new network structure for RPN and design our own R-CNN base network. In this experiment, we compare our solution with Faster R-CNN, denoted as *FR-CNN*. The training settings of reference schemes are the same with ours. At the inference phase, 10 RoIs of RPN proposals after NMS are reserved.

Table 5 compares several schemes in terms of detection performance. The AP (average precision) at different IoU thresholds are listed, e.g., AP0.7 means the AP value is calculated when the detected box is treated as true positive if IoU overlap with groundtruth box is greater than 0.7. There is an interesting phenomenon that Faster R-CNN with VGG performs almost the same as that with ZF when IoU threshold is 0.5. However, when the IoU threshold is increased, the advantage of VGG becomes visible. This suggests that shallow network structure can find the coarse location, but more powerful network can localize more precisely. We also change Faster R-CNN feature stride from 16 to 8, the performance dramatically improves. This shows that higher feature resolution is beneficial for license plate detection.

Comparing *Ours* and the *FR-CNN* series, we can easily find that our detector performs much better. The gain has three possible sources: 1) we use corner points as supervision while *FR-CNN ZF(or VGG)* uses bounding boxes as supervision; 2) we use a new network as the RPN base network and 3) the R-CNN in our network does not share feature with the RPN.

In order to figure out which design choices bring the gain and how much gain each design choice brings, we perform the following evaluation. *Ours RPN Only (Corner)* and *Ours RPN Only (BBox)* denote the methods which only use the RPN network as detector but the supervision information are groundtruth corner coordinates or bounding boxes, respectively. From Table 5, we can see that corner points can provide better supervision than bounding boxes and therefore achieve better performance. That is to say, regressing the corner points is a good option. If we check the AP values of *Ours RPN Only (BBox)* and *FR-CNN ZF(or VGG) S16*, we can observe that the *RPN only* solution outperforms original Faster R-CNN

Table 5: Comparison of detector performance

	AP0.5	AP0.6	AP0.7	AP0.8	AP0.9	Average
Ours RPN Only (BBox)	0.9416	0.9084	0.7973	0.5648	0.0758	0.6583
Ours RPN Only (Corner)	0.9434	0.9206	0.8476	0.6134	0.1317	0.6913
Ours	0.9668	**0.9640**	**0.9192**	**0.7572**	0.2994	**0.7813**
FR-CNN ZF S16	0.9039	0.8161	0.5827	0.3317	0.0694	0.5408
FR-CNN VGG S16	0.9075	0.8236	0.6021	0.3756	0.1159	0.5649
FR-CNN ZF S8	**0.9712**	0.9341	0.7897	0.6104	0.2258	0.7062
FR-CNN VGG S8	0.9709	0.9390	0.8043	0.6280	**0.3405**	0.7365

as well. This verifies that our RPN network is an effective structure. The gain obtained by separate feature representation for RPN and R-CNN is also notable. Comparing *Ours* with *Ours RPN Only (Corner)*, it can be observed that separate feature learning can achieve much better performance especially at higher IoU threshold. Therefore, using an independent light-weight CNN branch for finer local feature extraction from license patches is helpful.

The running time is evaluated using a NIVIDA K40 GPU. The inference speed in frames per second (fps) is given in Fig. 4. Using VGG as the base network for faster R-CNN achieves slightly better performance than using ZF, but both running time and model size increases dramatically. Our solution produces a much smaller model, whose size is only $1/57$ of the model size of fast R-CNN with VGG. Our network also runs 1.5 times faster than faster R-CNN with VGG S8. Although faster R-CNN with ZF runs faster, its performance is significantly inferior to our solution. Also note that our *RPN only* solution could run much faster, but we choose to sacrifice a little speed to get a much better performance. We believe that our choice strikes a good tradeoff between speed and performance for practical license plate detection applications.

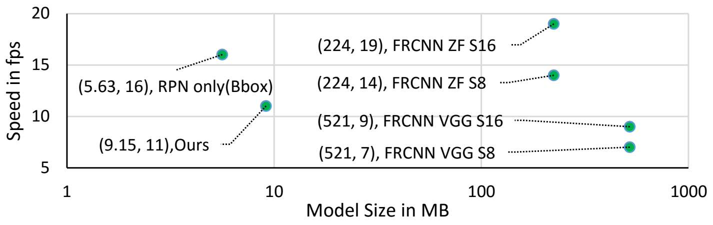

Figure 4: Runtime and Model Size of different schemes. The exact values of model size and speed in fps are given in the two-tuple beside each scheme.

5.2 Recognition Results Comparison

In the recognition part, we first rectify the input license patch and then crop character-level segments for recognition. The recognition networks for the last six characters share weights in order to make full use of the training data.

To validate the performance gain of each components, we carry out ablation study first. Table 6 shows the results. In the table, metric *Acc* denotes the accuracy of plate recognition, *Acc@1e* and *Acc@2e* mean the accuracies if we can tolerate 1 character error or 2 character errors, respectively. *Acc@1* denotes the accuracy of the first Chinese character and *Acc@2-7* mean the accuracy of the 2^{nd} to the 7^{th} characters. From the results, we can conclude that both rectification and weight sharing can bring obvious performance gain. In particular, rectification brings 3% and weight sharing brings 2.3% gain in accuracy. When rectification is enabled, *Acc@2-7* improves from 0.9611 to 0.9716 when we share weights (*Acc@1* does not change as weight sharing does not affect the first recognizer). When rectification is disabled, all the accuracy metrics decrease. This confirms that the rectification step is important, which in turn confirms that corner point regression in the detection stage is very helpful.

Table 6: Ablation study of recognition performance

Rectification	Shared Weights	Acc	Acc@1e	Acc@2e	Acc@1	Acc@2-7
✓	-	0.8678	0.9352	0.9632	**0.9545**	0.9611
-	✓	0.8599	0.9396	0.9746	0.9299	0.9674
✓	✓	**0.8905**	**0.9510**	**0.9755**	0.9501	**0.9716**

Fig. 5 visualizes the input patches, rectified patches and the segments generated by parallel STN subnetworks with or without rectification. The qualitative results show that our parallel STN subnetworks can perform satisfactory character segmentation even though it is trained only with the supervision of recognition groundtruth. However, if rectification is not enabled, training parallel STN subnetworks are much harder.

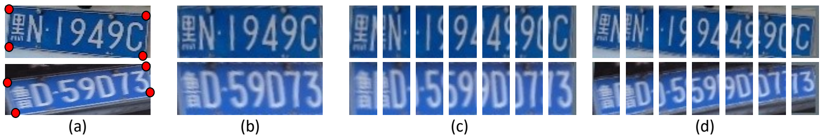

Figure 5: Visualize the outputs of Rectification and parallel STN subnetworks. (a) Detected patches and predicted corners (b) Rectified patches (c) Character segments produced by parallel STN subnetworks when rectification is enabled (d) Character segments when rectification is disabled

To compare our recognizer with the state-of-the-art recognizers, we implement two reference schemes. One is a character sequence encoding scheme proposed in [12], and the other is the work of H. Li et al. [15]. They treat license plate recognition as a sequence labeling task and leverage CNN and Bidirectional LSTM to solve this problem. The results are summarized in Table 7, where *CER* means character error rate.

Table 7: Comparison of recognition performance

	Acc	Acc@1e	Acc@2e	CER
[12]	0.7425	0.9036	0.9615	0.0614
[15]	0.3187	0.6743	0.8205	0.1961
ours	**0.8905**	**0.9510**	**0.9755**	**0.0314**

It is obvious that our recognition network outperforms both reference schemes. The CNN plus bidirectional LSTM solution performs the worst. This is because it is very sensitive to the distortion of license patches due to viewpoint changing and license plate number has very little correlation among all the characters. Therefore, such a solution is not suitable for license plate recognition in the wild. The character sequence encoding tries to predict each character based on the feature of entire plate, this scheme performs poorer than ours, because our recognition network can explicitly focus on pixel values of each character.

6 Conclusion

In this paper, we proposed a CNN-based ALPR system. In the detection stage, we adopt separate RPN and R-CNN for license plate detection and corner points regression. Rectification is performed to tackle different viewpoints of a license plate. In the recognition stage, parallel STN is leveraged for implicit character segmentation. We design an innovative shared-weight recognizer to make full use of the limited training data. Our solution outperforms the state-of-the-art detection and recognition schemes by a large margin. In the future, we plan to apply our solution to other types of license plates and evaluate its generalization capability.

References

[1] Christos-Nikolaos E Anagnostopoulos, Ioannis E Anagnostopoulos, Ioannis D Psoroulas, Vassili Loumos, and Eleftherios Kayafas. License plate recognition from still images and video sequences: A survey. *IEEE Transactions on intelligent transportation systems*, 9(3):377–391, 2008.

[2] Dong Chen, Gang Hua, Fang Wen, and Jian Sun. Supervised transformer network for efficient face detection. In *European Conference on Computer Vision*, pages 122–138. Springer, 2016.

[3] Konstantinos G Derpanis. Overview of the ransac algorithm. *Image Rochester NY*, 4 (1):2–3, 2010.

[4] Shan Du, Mahmoud Ibrahim, Mohamed Shehata, and Wael Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):311–325, 2013.

[5] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[8] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2315–2324, 2016.

[9] Tong He, Weilin Huang, Yu Qiao, and Jian Yao. Accurate text localization in natural image with cascaded convolutional text network. *arXiv preprint arXiv:1603.09423*, 2016.

[10] Tong He, Weilin Huang, Yu Qiao, and Jian Yao. Text-attentional convolutional neural network for scene text detection. *IEEE Transactions on Image Processing*, 25(6):2529–2541, 2016.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[12] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014.

[13] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[14] Chen-Yu Lee and Simon Osindero. Recursive recurrent nets with attention modeling for ocr in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2231–2239, 2016.

[15] Hui Li and Chunhua Shen. Reading car license plates using deep convolutional neural networks and lstms. *arXiv preprint arXiv:1601.05610*, 2016.

[16] Yi Li, Kaiming He, Jian Sun, et al. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387, 2016.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[18] Arik Poznanski and Lior Wolf. Cnn-n-gram for handwriting word recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2305–2314, 2016.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[22] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. Multi-oriented text detection with fully convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4159–4167, 2016.

[23] Zhuoyao Zhong, Lianwen Jin, Shuye Zhang, and Ziyong Feng. Deeptext: A unified framework for text proposal generation and text detection in natural images. *arXiv preprint arXiv:1605.07314*, 2016.