End-to-End License Plate Detection and Recognition Using Region Proposal Networks

Firas Naber
naber024@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

Li *et al* designed a deep neural network that can localize and recognize car license plates and characters simultaneously in one go. It is unique when compared to other approaches, where localization and recognition would be done in two separate steps. Doing both recognition and localization in the same network allows for time and cost efficiencies that are difficult to have if two separate networks were used instead. A region proposal network developed by Ren *et al.* was used as a base for the network. Extensive testing and experiments done by Li *et al.* proved promising results for their method. In this paper, the method proposed by Li *et al.* will be discussed thoroughly, with the main focus being on how the license plates would be found in an image through the detection network.

Keywords: deep neural networks, convolutional neural networks, region proposal networks

1 Introduction

Extracting the characters from images of license plates is an important concept in the modern world, specifically in intelligent transportation systems for reasons such as security and traffic control. There are many algorithms out there exclusively designed for the extraction of characters from license plates, and all of them require detecting (locating) the license plate in an image, and then recognizing (reading) the characters of that license plate.

Li *et al* [6] proposed a model structure that connects both license plate detection and recognition tasks and solves them simultaneously using a single neural network (**section 2.2**), unlike all other (known) license plate detection and recognition models, where they all do the recognition and detection tasks in two separate neural networks. A model that does both detection and recognition tasks simultaneously in a single neural network can provide efficiency in terms of time, cost, and other aspects as well. The model proposed by Li *et al* takes an image containing one license plate as its input and returns the location of the plate in the image, and recognizes its characters all at once, in a single end-to-end network. An overview of the structure of the model proposed by Li *et al* is shown in **Figure 1** [6].

The model in **Figure 1** contains many steps that work together to give the required output. In this paper, the main focus is going to be on discussing the left hand side of the model, i.e. starting from the input image and understanding how every step to its right works, all the way until the license plate detection network in **Figure 1**.

The upcoming sections of this paper will first give some background information on terms that will be used in the main part of the paper. Then, the paper will describe the method proposed by Li *et al.* and how it works, specifically the Region Proposal Network, Region of Interest (RoI) Integrating & Pooling, and the License Plate Detection Network parts seen in **Figure 1**. This paper then discusses some of the results provided by Li *et al.* Finally, a conclusion section is provided which summarizes everything discussed in this paper.

2 Background

There are a few terms and technologies that will be regularly mentioned in this paper. This section will briefly define and introduce those terms and technologies either in general or in the context of Region Proposal Networks (RPNs).

2.1 Loss Functions and Ground-truth

A loss function evaluates certain algorithms on how they model data assigned to them, by building an accurate predictor and comparing that predictor's results to the "ground-truth" results. If a loss function returned a large value (number), that would mean that the prediction value is far from the actual value.

There are several loss functions out there, however, all of them are categorized under two main loss types; classification losses and regression losses. Classification is used for predicting outputs from sets of discrete outputs, while regression is used for the prediction of continuous outputs (e.g. predicting changing cryptocurrrency prices) [8].

The term ground-truth will be used a few times in section 3. In the context of machine learning, ground-truth refers to checking the results of machine learning algorithms, such as Deep Neural Networks (DNNs) (**section 2.2.1**) against what is known in real life. Ground-truth consists of manually labeled and interpreted images where people have precisely

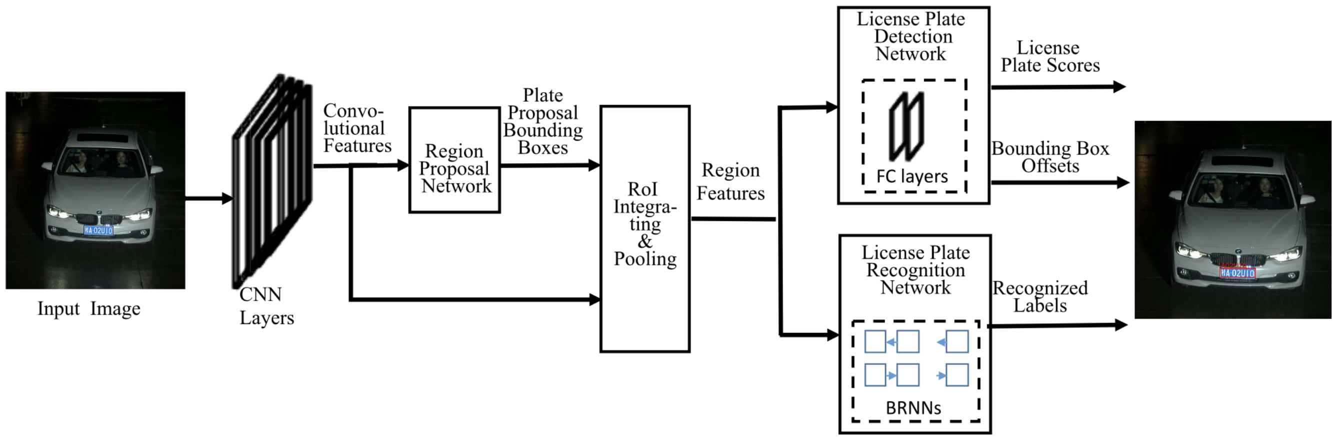

Figure 1. The structure of the model proposed by Li *et al.* which is made up of a region proposal network, RoI integrating & pooling, the license plate detection, and the license plate recognition layers [6].

identified where the license plates are and what characters they contain. [13].

2.2 Neural Networks

Neural networks are used to allow computer programs to recognize patterns and also solve problems in many fields such as artificial intelligence and machine learning.

2.2.1 Artificial Neural Networks. Neural networks, also known as artificial neural networks (ANNs) are a combination of nodes, or artificial neurons that connect together, forming node layers that each (neural network) contains an input, at least one hidden layer, and one output layer. Every connection between the nodes in a neural network has its own weight, and each connection in a neural network has its own value which is used to represent the strength of the connections between the nodes and layers, and are also necessary for training those neural networks.

Weights are also needed for forwarding the output of each node to the next layer. Each node has its own output value which would be multiplied by the weight vector value that connects that (output) node to the next one (input), sending the multiplication result to the next unit as its input. The process of only forwarding data (in one direction) from one layer of a neural network to the next one defines the network as a *feedforward* ANN. Usually, when a neural network is to contain many hidden layers, it would be classified as a DNN (**Figure 2**), and most DNNs are feedforward [4].

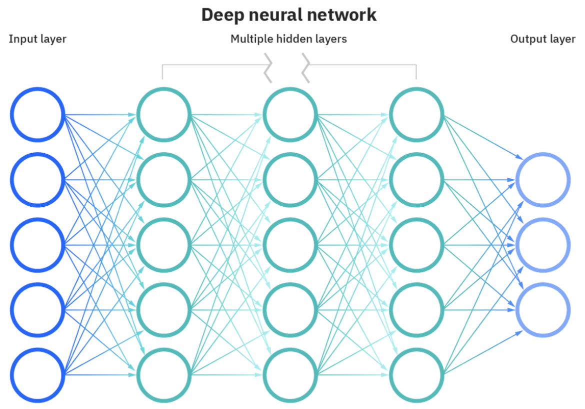

Figure 2. A simple deep neural network structure made of one input layer, three "hidden" layers, and one output layer [4].

2.2.2 Training Neural Networks. Training or optimizing neural networks is an essential task that has to be taken in order to prepare a neural network that has a high accuracy when being tested using input which has not been seen by the network before. For a neural network to be considered of high accuracy, the result of its loss function should be a small value, where the smaller its result is, the higher accuracy it would have.

Training would first be started by randomly initializing weights for the connections of the network. A neural network can be improved by switching its weights with others that would minimize the loss value found by the loss function used as much as possible [2].

2.3 Convolutional Neural Networks

Convolutional neural networks (CNNs) are an extension of ANNs. One main use of CNNs that distinguishes them from other types of neural networks is their superior performance with specific inputs such as images, which are used as the input in the model proposed by Li *et al.*

CNNs have three main types of layers; the convolutional layer (**section 2.3.1**), the pooling layer (**section 2.3.2**), and the fully-connected (FC) layer. The first layer in a CNN is always a convolutional layer. This layer can be followed by other convolutional or pooling layers, and the final layer in a convolutional network is a FC layer. FC layers usually take the output of the final convolutional or pooling layers as its input, which has to be *flattened* (**section 3.3**) first. The more layers there are in a CNN, the more complex it becomes [1][3].

2.3.1 Convolutional Layers. One common use of CNNs is to classify or identify one or more objects (features) of interest in an image input. This is done in the convolutional layers of the network by taking the input image in the form of numerical values and allowing the network to look for specific patterns. 3 components are required to accomplish that: an input, a filter (kernel), and a *feature* map. The input in our case would be a color (RGB) image with a license plate somewhere in it. An input RGB image is made from a matrix of 3-dimensions: a width, height, and the number of channels ($w \times h \times c$). 3 channels are normally found in RGB images: red, green, and blue, giving a c value of 3. The output of a convolutional layer is a feature map with the new dimensions $w' \times h' \times c'$, where the values w' and h' refer to the width and height of the map after applying the filter. c' is the output of the feature map which would increase when more convolutional layers are used. The output would then be exported into a 2-dimensional matrix to be used with the filter.

A simplified description of the filter's job is to detect features or specific elements in an input image by scanning through the image and checking whether those features are present or not. This process is called a convolution, which is where CNNs get their name from. Filters are 2-dimensional arrays of weights that can vary in size, however, they are most commonly 3×3 matrices. The filter would be applied to an area of the input image where the matrix values of the filter would be multiplied (dot product) by the values in the corresponding positions in the image. The dot product would then be sent to an output array. The filter would then shift, covering other matrices in the image where the entire process would be repeated. This is done over the entire image. The final product this process produces is what is known as a *feature map*. This process is visualized in **Figure 3** [3].

2.3.2 Pooling Layers. Pooling layers perform dimensionality reduction which decreases the number of parameters in the image inputs. Similarly to convolutional layers, they scan a filter across an input image and clusters the values using an aggregation function, giving an output array. Two common types of pooling are max-pooling and average-pooling. In max-pooling, the filter would move across the input image, given a reasonably configured stride (the number of pixels to shift over the input matrix) and sends the pixel with the largest value in the matrix to the output array. In average-pooling, the process is the same, however, the filter takes the average of the values it is placed on top of. Pooling is important because it helps reduce the complexity of a CNN and also improves its efficiency [3].

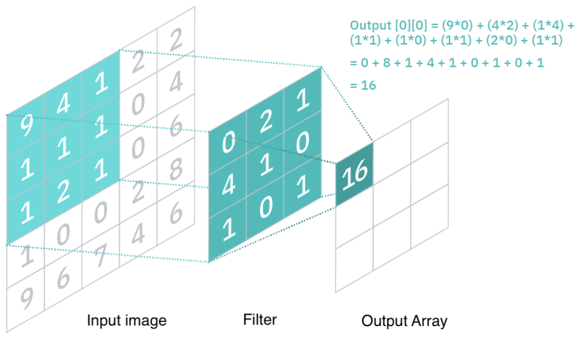
Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1*0) + (1*1) + (2*0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16
Input image
Filter
Output Array

Figure 3. A visualization of how the filter in the convolutional layer works including dot product (top right) [3].

2.4 Region Proposal Networks

The model used for the detection and recognition of license plates is based on the Region Proposal Network (RPN) model developed by Ren *et al* [9]. RPN is an algorithm that identifies certain objects in an image, and places candidate boxes around them. This is done by first generating "proposals" in the area where the identified objects are located. Proposals are objects detected by the model and are yet to be determined as wanted objects or not, which are license plates in Li *et al*'s paper. Proposals are generated by *sliding* or placing a network over a convolutional feature map provided by the CNN layers (**section 2.3**) such as the ones on the left in **Figure 1** [9].

The structure of the RPN (**Figure 4**) is made of a convolutional feature map (**section 2.3.1**) and a sliding window. A sliding window is an area selected on the input that is used to check for wanted objects (a convolutional filter). The central point of the sliding window is called its anchor where every anchor has its own scale and aspect ratio. The scale of an anchor is its size (width × height), while its aspect ratio is the width divided by the height. The dimensions of anchors are measured in pixels.

In Ren *et al*'s RPN, 3 scales and 3 aspect ratios were used, meaning that a total of 9 (3×3) proposals would be available for every pixel. Additionally, this means that there would be a total of $k = 9$ redundant anchors that overlap together.

The RPN has a classification layer and regression layer. The classification layer determines whether a proposal contains a wanted object or not using binary classification (0 if no object was found in the area, and 1 if a wanted object is found). The regression layer finds the coordinates of the anchors of the proposals and places bounding boxes around the areas with a binary score of 1, where bounding boxes are used to refer to areas of the proposed regions [9].

2.5 Intersection Over Union

Intersection over Union (IoU) is a statistic used for measuring the accuracy of object detectors. It is calculated by dividing the area of intersection by the area of union of the ground-truth bounding box (R_{gt}) and the detected bounding box (R_{det}) (**Equation 1**). IoU is used in loss functions to help measure how accurate the proposed bounding boxes are. [6][10].

$$\text{IoU} = \frac{area(R_{det} \cap R_{gt})}{area(R_{det} \cup R_{gt})} \tag{1}$$

3 Method

In this section, the method proposed by Li *et al.* will be described. The model uses 4 networks that are connected together to detect and recognize license plates. The purpose of the first 2 networks, RPN and RoI integration & pooling seen in **Figure 1**, is to provide an input to the license plate detection and the license plate recognition networks that output the license plate bounding boxes and their labels.

3.1 Plate Proposal Generation

3.1.1 RPN Training. Li *et al* modified the algorithm discussed in **section 2.4** to make it usable with car license plates. They designed 6 different scales with an aspect ratio (width/height) of 5, resulting in $k = 6$ anchors at each position of the input feature maps. Furthermore, the convolutional filter uses two 256-dimensional ($w' \times h' \times c'$, $c' = 256$) filters of sizes 5×3 and 3×1 (**Figure 4**) instead of the single square filter used in the RPN by Ren *et al.* The two filters in the RPN by Li *et al.* are placed simultaneously over the sliding positions. At this point, what was extracted from the image gets concatenated to form one 512-dimensional feature vector that is input into the classification and regression layers, similarly to the procedure done in **section 2.4** and seen in **Figure 4**.

The classification layer here outputs $2k$ scores (**Figure 4**) indicating the probabilities of the anchors either being license plates or not. Simultaneously with the classification layer, the regression layer outputs $4k$ values (**Figure 4**) referring to the offsets of the anchor boxes to a nearby ground-truth (**Figure 4**). The anchor boxes (before regression) have center points (x_a, y_a), and the width and height w_a and h_a respectively. After the 512-d vector passes through the regression layer, it gives the 4 scalar outputs (t_x, t_y, t_w, and t_h), where t_x and t_y refer to the scale-invariant translation and t_w and t_h refer to the log-space height/width shift, that are used to transform to ground-truth boxes by allowing the computer to compare input license plate images together (refer to Singh and Davis [11] for details).

The bounding box offsets are given by

$$x = x_a + t_w w_a, y = y_a + t_y h_a,$$
$$w = w_a \exp(t_w), h = h_a \exp(t_h),$$

where x and y are the center coordinates of the bounding box after regression and w and h represent the width and height, after regression as well, respectively.

For a convolutional feature map of dimensions $M \times N$, there would be $M \times N \times k$ anchors, where a lot of overlapping is to be expected between them. Inaccurate and biased RPN training would occur if the number of negative anchors (anchors that do not contain license plates and are given the binary value 0) used for training was greater than that of the positive anchors (anchors that contain the target object and are given the binary value 1), which in our case are license plates. Li *et al.* randomly sampled 256 anchors (of size $M \times N \times k$) from one image as a mini-batch (user-specified number of training items), with a 1:1 ratio between the number of positive and negative anchors, to avoid having more negative than positive anchors.

Whether an anchor is positive or negative is determined, during training is based on its IoU score (**section 2.5**). If the IoU score for an anchor is greater than 0.7, they would be considered as positive, while an anchor with an IoU score of less than 0.3 would be a negative, where anchors with IoU values between 0.3 and 0.7 would not be used, unless none of the anchors available have a value that is greater than 0.7, where the greatest of them would be used and considered positive. In a case where there aren't enough positive anchors, they get merged with the negative ones giving more positive anchors to be utilized [6][9].

The binary logistic loss function (predicts either 0 or 1) is used by the classifier layer, and a loss function known as smooth L_1 (a loss function specifically used in object detection systems) is employed in the regression layer (refer to Ren *et al.* for more details about smooth L_1 loss).

$$L_{RPN} = \frac{1}{N_{cls}} \sum_{i=1}^{N_{cls}} L_{cls}(p_i, p_i^\star) + \frac{1}{N_{reg}} \sum_{i=1}^{N_{reg}} L_{reg}(t_i, t_i^\star) \tag{2}$$

N_{cls} refers to the size of a mini-batch and N_{reg} refers exclusively to the number of positive anchors in this batch [6]. L_{cls} refers to the binary logistic function used while L_{reg} refers to the smooth L_1 loss function. p_i refers to the probability that i is a license plate and $p_i^\star$ is a binary value of either 1 (positive anchor) or 0 (negative anchor). t_i refers to the predicted coordinates for the anchor i, which are the center of the anchor and its width and height, while $t_i^\star$ refers to the related offsets for anchor i relative to the ground-truth (**section 2.1**).

3.1.2 RPN Testing. Non-Maximum Suppression (NMS) (an algorithm that would select the best bounding box out of

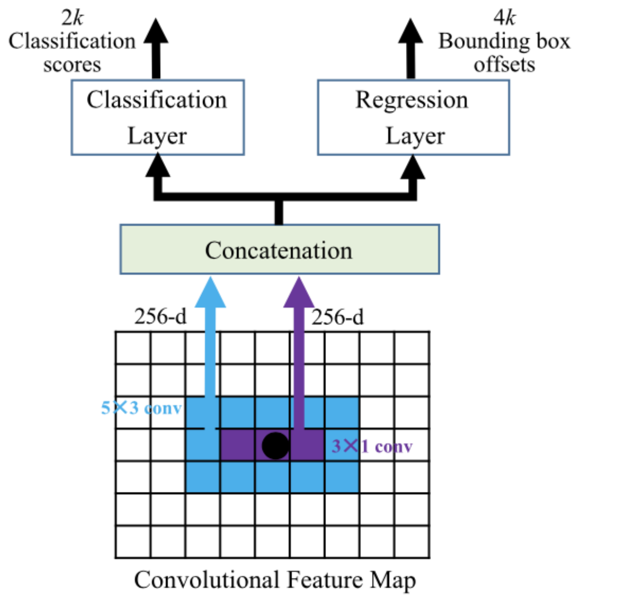

Figure 4. Plate proposal generation using the altered RPN by Li *et al.* [6].

a set of overlapping (IoU) boxes on an image) would be employed on the entire network when testing the RPN to select 100 proposals for the processing and pooling procedures of the RPN in **section 3.2** [6][12].

3.2 Integration and Pooling

The 256 anchors sampled from $M \times N \times k$ anchors earlier on in **section 3.1** undergo bounding box regression where they would be used in later steps for plate recognition and detection. Those bounding box samples are defined as

$$p = \left(x^{(1)}, y^{(1)}, x^{(2)}, y^{(2)}\right),$$

where $\left(x^{(1)}, y^{(1)}\right)$ corresponds to the left top coordinate of the bounding box, and the $\left(x^{(2)}, y^{(2)}\right)$ is for the right bottom coordinate.

Positive proposals are defined as $p_{i,j} = \left(x_{i,j}^{(1)}, y_{i,j}^{(1)}, x_{i,j}^{(2)}, y_{i,j}^{(2)}\right)$ for $i = 1, ..., n$. For every positive proposal associated with the same ground-truth license plate g_j, a bounding box with bigger size would be created (the red box in **Figure 5**) by merging the positive proposals together. This is done to contain all the characters in the plate since some positive anchors are unable to cover all the characters on the plate (the green box in **Figure 5**). Those new bounding boxes b_j for $j = 1, ..., m$ would then be used as positive image samples later on for plate detection and recognition. This process is only implemented in the training stage, where the 100 proposals selected in **section 3.1.2** with higher plate classification scores are processed directly for detection and recognition.

Since the proposals were sampled (from $M \times N \times k$ anchors), there is a chance of having many negative anchors. Therefore, any bias due to the uneven ratio of positive and negative anchors has to be reduced. This is done for the purpose of training, by randomly selecting $3m$ negative anchors from the 256 samples and also choosing another $4m$ random samples (positive or negative) and form a mini-batch. Regions of interest (RoI) max-pooling would then be performed to get region features (the output of the RoI layer) of the fixed size 28×4 (width $\times$ height).

Having fixed size feature maps is essential, because the plate detection and recognition network layers to be used in the next steps can only take one size (to be used for testing the model by Li *et al.* in **section 4**) as an input at once during training and testing, which is also why the RoI integrating and pooling step is used in the first place in the model proposed by Li *et al.* [6].

3.3 License Plate Detection Network

Here is where the model by Li *et al.* decides whether the proposed RoIs are license plates or not, and also where the plate bounding boxes are refined. First of all, two FC layers (**section 2.3**) are needed in order to extract determining features from the feature maps input into the license plate detection network. Those features (in the form of a matrix) would first be flattened or transformed into a 1-dimensional vector, allowing them to fit as inputs in the two vertically structured FC layers, which have a similar structure to that of the input layer in **Figure 2**.

The outputs of the two FC layers would be input into two other transformation layers, where license plate classification would take place, and bounding-box regression as well. The classification transformation layer gives 2 outputs which are the probability (between 0 and 1) of the RoIs being license plates or not, and the regression layer gives the final bounding-box coordinate offsets of all the proposals [6].

3.4 License Plate Recognition Network

The region features from the RoI pooling and integration layer would be input into the license plate recognition network as well, simultaneously with the license plate detection network. Bi-directional recurrent neural networks (BRNNs). They connect two hidden layers of opposite directions in an ANN together, and a type of neural networks known as connectionist temporal classification (CTC) are used for license plate character sequence decoding, outputting recognizable license plate labels (plate numbers).

A sequence labelling algorithm (mapping sequences algorithmically) is used so that an issue known as character segmentation would be prevented from happening. Character segmentation is when an image with license plate characters would be segmented into sub-images each with one character, which can be problematic if some characters were

Method	Detection Performance Average (%)	Detection Speed Per Image (ms)	End-to-end Speed Per Image (ms)
Zhou *et al.* [16]	90.22	475	-
Li *et al.* (2013) [5]	91.52	672	-
Yuan *et al.* [15]	97.69	**42**	-
Li *et al.* (2019) [Detection Only] [6]	99.51	283	-
Li (2019) *et al.* [Jointly-trained] [6]	**99.73**	279	310

Table 1. Testing results for the model proposed by Li *et al.* (2019) using the PKUData dataset [6].

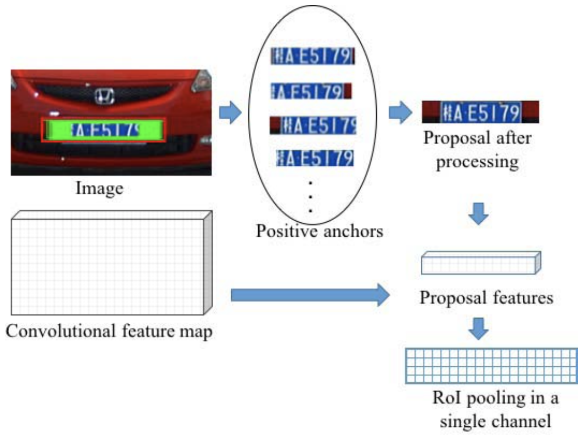
Image
Positive anchors
Proposal after processing
Convolutional feature map
Proposal features
RoI pooling in a single channel

Figure 5. The procedure steps taken in proposal processing and pooling training [9].

incorrectly segmented, meaning that they might not get correctly recognized [7]. Character segmentation happens here due to every character of the license plates being independent. BRNNs are able to give an output of feedforward and feedback (the opposite of feedforward networks) simultaneously. BRNNs are known for their good performance with letter and character recognition.

More in-depth details about the license plate recognition network can be found in the paper by Li *et al.* [6][14].

4 Results

Li *et al.* used 4 datasets to test their model:

- Car license plates from China, known as CarFlag-Large.
- Application-Oriented License Plate (AOLP).
- Caltech-cars 1999.
- PKUData by Yuan *et al.* [15].

The focus in this paper will be on the results provided from testing done using the image dataset PKUData by Yuan *et al.* found in **Table 1**. The method proposed by Li *et al.* was compared to another 3 "state-of-the-art" plate detection methods as they call them in their paper. 5 different undisclosed photographing conditions (e.g. weather or time of day when the images were captured) were used to test and train their method. The average detection performance (accuracy) results of the different capturing conditions images was then taken and used for comparisons.

Training the model was done using 322,000 images from the CarFlag-Large dataset. Training is done using this dataset because both it and the PKUData images contain only Chinese license plates. Again, testing the dataset would be done using the PKUData dataset of 3,977 images containing Chinese license plates somewhere in them.

The results show that the network achieves an average detection rate of 97.33%. This is 2% greater than the previous highest average from all the other methods used for the comparison. The method by Yuan *et al.* has a higher detection speed (the speed in milliseconds at which license plates would be detected) since it is using support vector machines (SVMs); a different approach to classification, regression and outlier detection, instead of DNNs. The model by Li *et al.* comes in second place with a detection speed of 279ms, and is also the only method that integrates both detection and recognition losses in one network, with an end-to-end detection and recognition speed of 310ms, which as discussed earlier, has time, cost, and other efficiency advantages [6].

The testing done on the 3 other datasets gave similar results to those from the PKUData dataset. This shows that the results from the tests done are fairly accurate and can tell us that the model proposed by Li *et al.* has a good potential of being used or further improved to be used in the real world.

5 Conclusion

Li *et al.* proposed a DNN for the detection and recognition of license plates that does both of the steps simultaneously, in a single pass, with higher accuracy and efficiency than that of other known previous methods available. In this paper, the main focus was on how RPN works and outputs bounding boxes, where RoI integration and max-pooling is performed on them, giving region features as their output. Testing done on the network gave good results, out-performing most other license plate detection methods.

Acknowledgments

I would like to thank Kirbie Dramdahl, Nic McPhee, and Elena Machkasova for their advice, feedback, and dedication to helping with this paper. I would also like to thank my friends and family for all their help and support.

References

[1] Arc. 2018. Convolutional Neural Network. https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05

[2] Vitaly Bushaev. 2017. How do we 'train' neural networks? https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73

[3] IBM Cloud Education. 2020. What are convolutional neural networks? https://www.ibm.com/cloud/learn/convolutional-neural-networks [Online; accessed 21-Oct-2021].

[4] IBM Cloud Education. 2020. What are neural networks? https://www.ibm.com/cloud/learn/neural-networks [Online; accessed 20-Oct-2021].

[5] Bo Li, Bin Tian, Ye Li, and Ding Wen. 2013. Component-Based License Plate Detection Using Conditional Random Field Model. *IEEE Transactions on Intelligent Transportation Systems* 14, 4 (2013), 1690–1699. https://doi.org/10.1109/TITS.2013.2267054

[6] Hui Li, Peng Wang, and Chunhua Shen. 2019. Toward End-to-End Car License Plate Detection and Recognition With Deep Neural Networks. *IEEE Transactions on Intelligent Transportation Systems* 20, 3 (2019), 1126–1136. https://doi.org/10.1109/TITS.2018.2847291

[7] Y. Lu. 1993. On the segmentation of touching characters. , 440-443 pages. https://doi.org/10.1109/ICDAR.1993.395699

[8] Ravindra Parmar. 2018. Common Loss functions in machine learning. https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23 [Online; accessed 21-Oct-2021].

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 6 (2017), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

[10] Adrian Rosebrock. 2017. Intersection over Union (IoU) for object detection. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ [Online; accessed 7-October-2021].

[11] Bharat Singh and Larry S Davis. 2018. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3578–3587.

[12] Vineeth S Subramanyam. 2021. Non Max Suppression (NMS). https://medium.com/analytics-vidhya/non-max-suppression-nms-6623e6572536

[13] Techopedia. [n.d.]. Ground Truth. https://www.techopedia.com/definition/32514/ground-truth [Online; accessed 21-Oct-2021].

[14] Wikipedia contributors. 2021. Bidirectional recurrent neural networks — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Bidirectional_recurrent_neural_networks&oldid=1052842662 [Online; accessed 29-November-2021].

[15] Yule Yuan, Wenbin Zou, Yong Zhao, Xinan Wang, Xuefeng Hu, and Nikos Komodakis. 2017. A Robust and Efficient Approach to License Plate Detection. *IEEE Transactions on Image Processing* 26, 3 (2017), 1102–1114. https://doi.org/10.1109/TIP.2016.2631901

[16] Wengang Zhou, Houqiang Li, Yijuan Lu, and Qi Tian. 2012. Principal Visual Word Discovery for Automatic License Plate Detection. *IEEE Transactions on Image Processing* 21, 9 (2012), 4269–4279. https://doi.org/10.1109/TIP.2012.2199506