

# Detection-Recognition Networks for Car License Plates

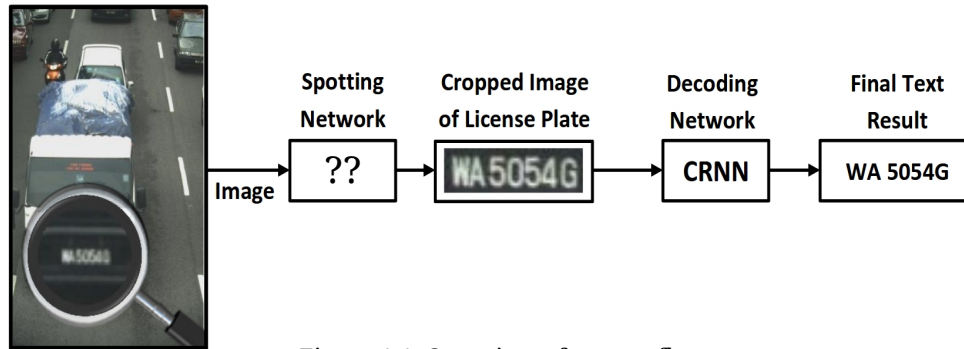


Figure 1.1, Overview of system flow

A complete License Plate Detection-Recognition (LPDR) system is typically consist of two components:

1. Plate Detection: means to spot or localize the license plate in natural image and generate **suitable bounding boxes (cropped image)**. In this part, the CTPN[1] and TextBoxes[2] Models had purposed previously, however, both of them have shown unacceptable results, **so we are using them for preparing test-set for the recognition part, rather than manual cropping method.** For example “CTPN” has successfully cropped 1438 images from Jakarta dataset, while 362 images have manually cropped. However, we still haven't start working on the plate detection part. Moreover there are some observations that we need to consider in this part, if the input image as shown in Figure 1.1 then the suitable bounding box ”cropped image” of detection should be as shown in Figure 1.3 not as Figure 1.2 to obtain an acceptable result from the character recognition method. “Reason beyond unsuccessful recognition by CRNN for Figure 1.2 will be clear in section 2”.



Figure 1.1

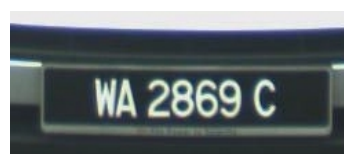


Figure 1.2



Figure 1.3

2. Plate Recognition: aims to identify the characters depicted within the bounding boxes or cropped image. There are two different methods on plate recognition. The first one performs character segmentation firstly and then recognizes each character. The other one regards the character string recognition as a sequence labeling problem, and recognizes all characters in the license plate one-off. This method avoids the challenging task of character segmentation. Furthermore, all recent text recognition consider this problem as a sequence labeling problem. So we will introduce the second method that using Convolutional Recurrent Neural Network (CRNN)[3] for plate character recognition.

### 2.1 CRNN Architecture :



Figure 2.1, Network architecture

The network architecture of CRNN, as shown in Fig. 2.1, consists of three components, including the convolutional layers, the recurrent layers, and a transcription layer, from left to right.

#### 2.1.1 Convolutional Neural Network "CNN" :

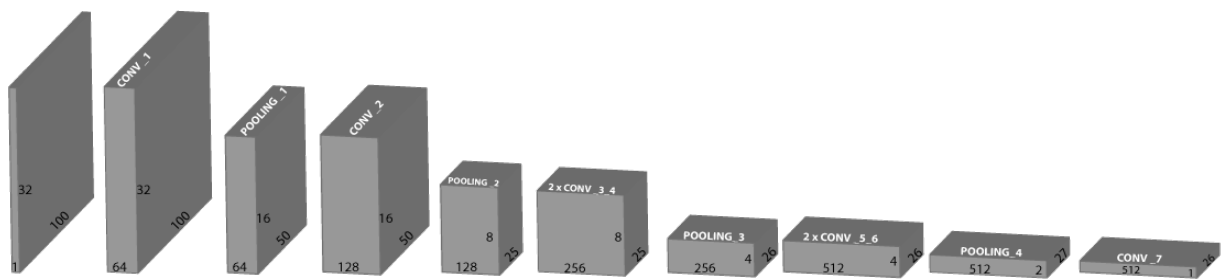


Figure 2.1.1, Convolutional Neural Network Architecture

According to Figure 2.1.1, Input image is resized to 32x100 and converted to gray-scale, so we got an input dimension of 32x100x1. This network consists of 7 convolutional layers and 4 maxpooling layers. It is based on VGG-VeryDeep architecture[4]. VGG is adopted only 3x3 convolution and 2x2 pooling throughout the whole CNN "with different stride and padding". VGG also shows that the depth of the network plays an important role. The output dimension of the CNN is 1x26x512. Which will be mapped to sequence as 26 x 512 . where 26 are the number of frames in image and 512 are feature extracted. This output is mapped to the Bidirectional Recurrent Neural Network "BLSTM" as shown in Figure 2.1.2.

### 2.1.2 Recurrent Neural Network “RNN”:

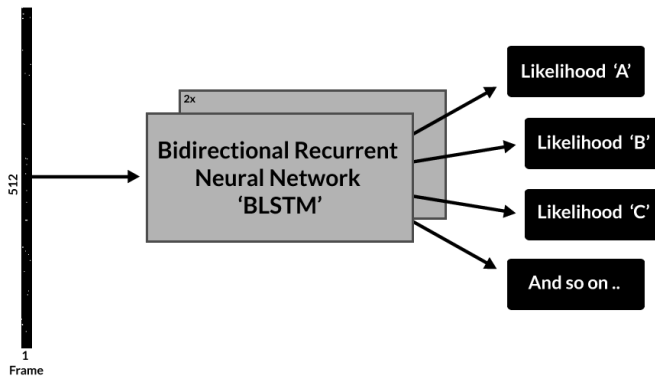


Figure 2.1.2.1

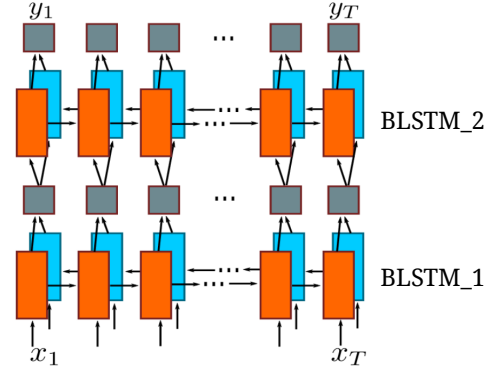


Figure 2.1.2.2

It contains a memory cell and three multiplicative gates, which can store the contexts for a long period of time, and capture long-range dependencies between sequential features. For our task of character string recognition, it would be helpful to have access to the context both in the past and in the future. So bidirectional-Long Short Term Memory (BLSTM) is applied here. Our network has two BLSTM, each BLSTM has two separated hidden layers, one of which processes the feature sequence forward, while the other one processes it backward. For each hidden layer, all LSTMs share the same parameters. Both hidden layers are connected to the same output layer, providing it with information in both directions along the input sequence as shown in Figure 2.1.2.2. BRNN's output has dimension of  $37 \times 26$  where 37 is representing number of output classes “<blank> 0-9 A-Z” and 26 is representing number of frames.

### 2.1.3 Connectionist Temporal Classification “CTC”:

Connectionist Temporal Classification is a loss function that useful for performing supervised learning on sequence data, without needing an alignment between input data and labels “no segmentation required”. It is the last competent of our network that transform the sequence of probability estimated by RNN into character string. It is applied to the output layer of RNN. It also remove the duplicated character from successive frames. “The formulation of the conditional probability is briefly described as follows: The input is a sequence  $y = y_1, \dots, y_T$  where  $T$  is the sequence length “26”. Here, each  $y_t \in <|L'|>$  is a probability distribution over the set  $L' = L \cup \cdot$ , where  $L$  contains all labels in the task (0-9 A-Z), as well as a 'blank' label denoted by  $\cdot$ . A sequence-to-sequence mapping function  $B$  is defined on sequence  $\pi \in L'^T$ , where  $T$  is the length.  $B$  maps  $\pi$  onto  $l$  “label sequence” by firstly removing the repeated labels, then removing the <blanks>. For example,  $B$  maps “--hh-e-l-ll-oo--” ( $\cdot$  represents 'blank') onto “hello”. Then, the conditional probability is defined as the sum of probabilities of all  $\pi$  that are mapped by  $B$  onto  $l$ :

$$p(l|y) = \sum_{\pi: B(\pi)=l} p(\pi|y), \quad (1)$$

where  $p(\pi|y) = \prod_{t=1}^T y_{\pi_t}^t$ ,  $y_{\pi_t}^t$  is the probability of having label  $\pi_t$  at timestamp  $t$ . Directly computing Eq. 1 would be computationally infeasible due to the exponentially large number of summation items. However, Eq. 1 can be efficiently computed using the forward-backward algorithm described in [5] [3].

### 3.0 Preparation of Training datasets:

Two training datasets, generated using python PIL library, were used to fine-tune the trained CRNN. The summary of the generated dataset is as shown in Table 3.1.

Table 3.1, Comparison between previous created and recently generated training-sets.

	Training-set #1	Training-set #2
No. of images	500,000	110,000
Character/Image	5 – 7	12 – 16
Image dimension	32 x 100	various dimension (font size change according to the number of character in the image)

We noticed that when the CRNN is trained on the generated image (dataset #1) that contain 5 – 7 letters with fixed dimension of 32 x 100, it is performing well on the image that has same or near aspect ratio of the dimension (32 x 100 pixels) but perform not so well with the Jakarta dataset. When carefully revisit with the Jakarta dataset again we noticed that the aspect ratio between its height and width is far from the 5 – 7 letters with 32 x 100 pixels. *The width of the Jakarta dataset is 5–6 times of the 32 pixels height.* So new training-set, training-set #2 with 110,000 images, were generated with different dimension and character distribution as stated in the right hand side table. The below table will give an explanation of training image size effect.

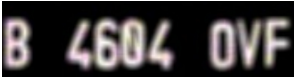
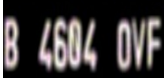


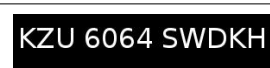

Input image to CRNN (Jakarta license plate )		Jakarta License plates has different aspect ratio between its width and height because its contain 8 characters and 4 spaces , so thats make it width larger than normal license plates “Like Malaysia”, So after the image is resized, it is lose some of character shape “feature” and become unable to recognize some character correctly “Like G and C ” , because it has previously trained on perfect width and height that fit 32 x 100 exactly “so the letter hasn’t had any change in its shape”.
Resizing of input image to 32 x 100		
Sample of Training-set #1		
Resizing of Training-set #1 to 32 x 100		
Sample of Training-set #2		
Resizing of Training-set #2 to 32 x 100		






Figure 3.0.2: Samples of Training-set #1

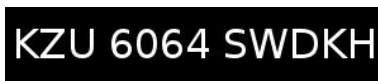



Figure 3.0.3: Samples of Training-set #2

Character	No. in Training-set
0	38050
1	42619
2	38494
3	38324
4	60140
5	38108
6	81029
7	68361
8	59446
9	55527
A	25079
B	25522
C	25473
D	25388
E	25339
F	25413
G	25166
H	25471
I	0
J	25545
K	25240
L	25408
M	25465
N	25447
O	0
P	25432
Q	25456
R	25306
S	25290
T	25642
U	25346
V	25403
W	25394
X	25585
Y	25177
Z	25205

#### 4.0 Recognition Results of Jakarta Dataset:

By performing a test over the given 1800 license plates on the Jakarta dataset, we obtained the following results:

Edit Distance	0	1
Number of image	1793	7

Percentage Perfect	Average Edit Distance
99.61	0.0039

Now, one successful recognition sample and two failure samples will be presented with more explanation with there image, output visualization, and predicted label. The output visualization in the figure below shows the predicted label “character” per each frame with its confidence. ( x-axis is represent the 26 frames while the y-axis represent 37 classes “ <blank> 0-9 A-Z ” )

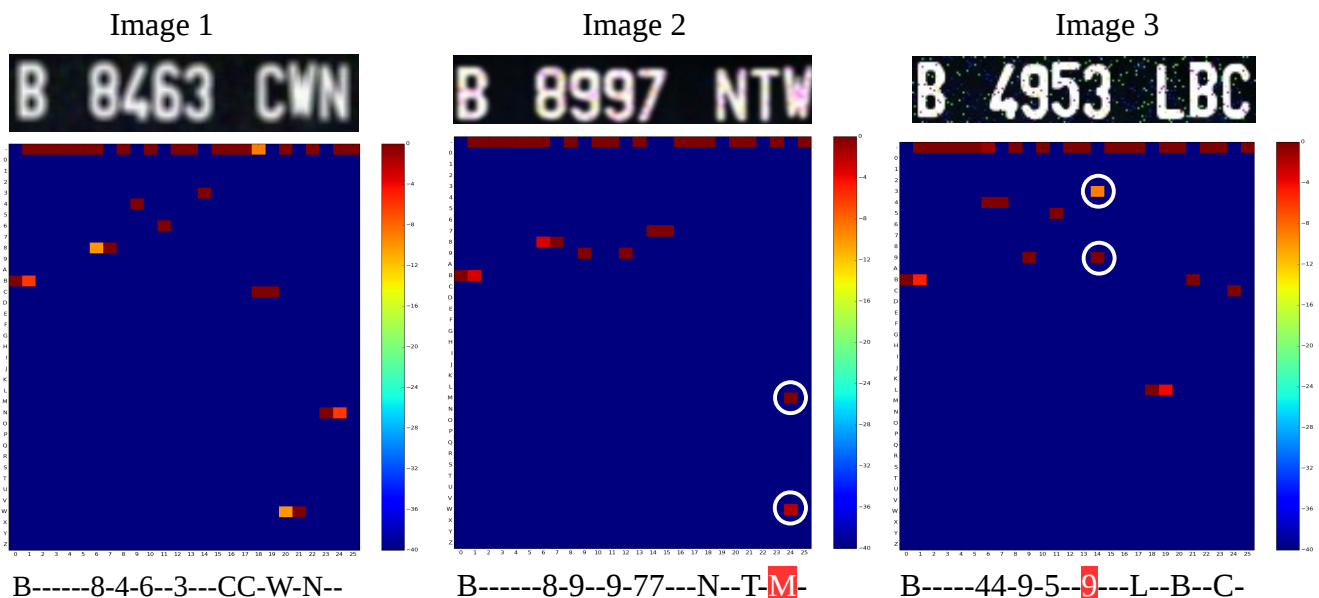


Image 1 has successfully recognized to the corresponding output from the RNN (B-----8-4-6--3---CC-W-N--) then CTC will form this output as (B8463CWN) “remove the duplicated character from successive frames”.

Image 2 has edit distance of 1 because the frame #25 has recognized as 'M' rather than 'W', however , we can notice that the letter 'W' has second confidence after confidence of 'M' directly.

Image 3 has edit distance of 1 because the frame #15 has recognized as '9' rather than '3' because of the noise in the image, however , we can notice that the number '3' has second confidence after confidence of '9' directly.

Input Image	B 367 OPR	B 1838 GEX	B 4604 OVF	B 4228 OVE	B 136 OKC
Inference	B367DPR	B1838GEY	B4604DVF	B4228DVE	B136DKC

Last 5 Images that has been unsuccessfully recognized

References:

- [1] Shi, B.; Bai, X.; and Yao, C. 2015. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition.
- [2] M.Liao, B.Shi, X.Bai, X.Wang, W.Liu (2015), TextBoxes: A Fast Text Detector With a Single Deep Neural Network.
- [3] Zhi Tian, Weilian Huang, Tong He, Pan He and Yu Qiao (2016) Detecting Text in Natural Image with Connectionist Text Proposal Network.
- [4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large scale image recognition. CoRR, abs/1409.1556, 2014.
- [5] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In ICML, 2006.