Android-Based License Plate Recognition using Pre-trained Neural Network

Rizqi K. Romadhon, Muhammad Ilham, Nofan I. Munawar, Sofyan Tan, and Rinda Hedwig

Abstract— **In this research an automobile license plate recognizer is developed in a mobile device based on the android platform to be used as an alternative solution to record road-side transactions, such as parking and fine. The license plate recognition is implemented using artificial neural network, where training of the neural network is carried out in a desktop computer instead of mobile device to speed up the training using back propagation algorithm. Result of the training is the artificial neural network along with their weight is implemented in the mobile device to recognize characters in license plate. The training result showed that the neural network is able to recognize 88.2% characters in sample images outside the training set. The image processing and neural network implemented in the mobile device managed to recognize an average of 71% characters in sample license plate images which are categorized as having good image quality. Automobile license plate information is also saved in database inside the mobile device.**

Index Terms— **android-base, back-propagation algorithm, license plate recognition, neural network, pre-trained**

I. Introduction

ANDROID-based image recognition is one of the applications that are used in many mobile devices that are widely available nowadays. As an open source, android is favorable and is applicable in image recognition as well. In many parts of Jakarta city, the parking lots are along the street and monitoring cars come and go is not easy as well as for security purposes. A mobile device will be a good choice as one of the media for the parking staff to capture license plate; later on the image will be recognized and saved in the device. License plate recognition is widely used in many automatic parking systems in many countries and widely reported in series of papers [1-10]. While in this paper we use a simple back-propagation algorithm for license plate recognition. This simple android-based application will benefit parking staff in monitoring the cars along the street of Jakarta.

Manuscript received July 27, 2012. This work was supported in part by the Computer Engineering Laboratory.

R. K. Romadhon was with the Computer Engineering Department, Binus University, Jakarta, Indonesia (e-mail: rizqi_batavia @yahoo.com).
M. Ilham was with the Computer Engineering Department, Binus University, Jakarta, Indonesia (e-mail: cr7_mu35@yahoo.com).
N. I. Munawar was with the Computer Engineering Department, Binus University, Jakarta, Indonesia (e-mail: novan_irkham@yahoo.com).
S. Tan is with the Computer Engineering Department, Binus University, Jakarta, Indonesia (phone: +6221-534-5830 ext. 2205; e-mail: sofyan@binus.edu).
R. Hedwig is with the Computer Engineering Department, Binus University, Jakarta, Indonesia (phone: +6221-534-5830 ext. 2205; e-mail: rinda@binus.edu).

II. Literature Review

License Plate Recognition (LPR) has been actively studied for couple of decades and still is an active subject of research because of its wide use and potential application [1]. One of the most popular approaches for License Plate recognition is using artificial neural networks (ANN). Various approaches using neural network can be seen in [1-4]. Variation of the ANN used for LPR can been in for example in [1-2]. In [1] a Radial Basis Function Neural Network (RBFNN) is used instead of the more popular Back-Propagation (BP) neural network. The evaluation showed better performance over BP neural network. Other researcher [2] improved the BP learning algorithm to increase recognition rate. In general there are many possible algorithms using ANN to recognize characters in license plate. This shows that the ANN is still a popular choice to implement LPR. The benefit of using ANN for real-time applications is because the training and classification stage can be separated, where the classification time is much faster than the training time. The classification of image feature can thus be implemented in mobile devices with low computing power.

Some researchers have even implemented the LPR algorithm into Android device, such as in [3-4]. They include image processing to detect the position of the license plate in the capture image. Such approach can be time consuming in low computing power devices such as Android devices. In this paper the license plate detection problem is omitted simply by having the user to point the camera directly in front of the license plate with visual guide in the graphical user interface. The mobile device then just have to perform light image processing to remove noise and segmentation of the characters before feeding them into the feed-forward neural network.

III. Pre-Training the Neural Network

The neural network is trained to recognize a character in a 14x10 pixels-image containing a single character scaled to fill the whole image area. Training is carried out in a desktop computer to speed-up the process. There are 4 image samples for each numeric and upper-case alphabet characters, hence 144 image samples are used as the training set for the neural network. Each sample image is preprocessed manually by cropping a character from a larger image, converting to B&W, and resizing it to 14x10 pixels. Each larger image is a photograph of a real automobile license plate. The authors chose 14x10 pixel input image to minimize the calculation cost when the neural network is implemented in an android mobile device.

MATLAB® is used in the desktop computer to preprocess image samples, construct and train the neural network [11]. After a character image is manually cropped from a larger license plate image, it is converted from RGB format to grayscale and then converted to black and white binary data, where the threshold is calculated using Otsu algorithm [12-13]. Each sample image is further scaled down to 14 pixels tall and 10 pixels wide, suitable for the input layer of the neural network. The whole 144 binary image samples are combined into a training set.

The architecture of the neural network is a multilayer feed forward neural network, where two hidden layers are used to model the character recognition behavior as shown in figure 1. The input layer is consisted of 140 nodes, each connected to one pixel of the 14x10 input character images. The output layer contains 36 nodes, each for one of 36 numerical and alphabet characters (0 to 9 and A to Z). The first hidden layer contains 44 nodes, while the second contains 36 nodes. Output of all nodes in the hidden and output layers are calculated using the sigmoid activation function (1) to produce continuous output function with an output range between 0 and 1.

$$f(x) = \frac{1}{1+e^{-x}} \tag{1}$$

After training the neural network using back-propagation algorithm in MATLAB®, the resulting weights matrix is exported to be used in the mobile device as a fix weights matrix implemented in the same neural network architecture, but without the capability to retrain the weights.

IV. Implementation in Android-based Mobile Device

Most android device includes camera that can be used to capture a license plate as a source image, however the source image must be automatically preprocessed into 14x10-pixels character images for recognition by the neural network. The manual cropping used during training is not viable in the implementation. Therefore a light image preprocessing algorithm is developed in the mobile device to resize, convert, and segment the source image into 14x10-pixels character images.

The first step in the image preprocessing is pre-resizing the source image into lower resolution to improve performance and simply cropping out part of the source image that is not a license plate number. Assuming that the aspect ratio of any standard license plate is constant, the user can be guided to aim the camera at a certain distance so that the license plate will completely fill a certain area in the electronic view finder on the mobile device. The application can then easily discard all pixels outside the predefined area and size down the image into 512x150 pixels for faster processing, while maintaining most of the image details. Registration date information on the plate is discarded in this pre-resizing step because of their lower position in the plate.

Color information is not used in the image processing, therefore the pre-resized source image is converted to grayscale and then into black and white binary source image for faster image processing.

The next step is to segment the binary source image into multiple images each containing a single character. It is assumed that all characters in the license plate is separated from each other, therefore segmentation can be done by uniquely labeling each island of connected pixels as a single character. Connected pixels are search in the 8-neighbor of each pixel as seen in figure 2. Figure 3 shows islands that contain pixels less than a particular threshold number are considered as noise and thus neglected. The big islands are cropped out by finding the left, right, upper, and lower edge coordinates of the islands in the source image. These cropped out islands are then scaled and resized to 14x10 character images.

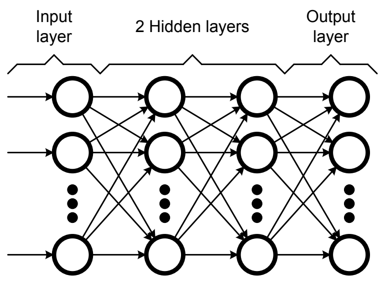


Fig. 1. The 144 input nodes are directly fed from the 14x10 pixels of the input character image, and then fully connected to the two hidden layers containing 44 and 36 nodes respectively and finally to the 36 nodes of the output layer.

N1	N2	N3
N4		N5
N6	N7	N8

Fig 2. The 8 neighbor pixels are searched for any connected pixels with binary intensity of one.

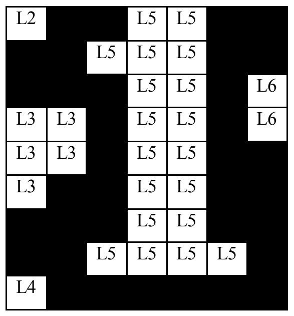


Fig 3. In a low resolution example, five islands of connected pixels are found in the above image, where one biggest island (labeled L5) is the character image, while the other islands are considered as noise.

The same neural network architecture as those used in the pre-training is implemented in the mobile device. Weights matrix from the pre-training process is imported into the neural network in the mobile device, resulting in exactly the same network behavior as the pre-trained neural network. Character images resulted from image preprocessing in the mobile device are then fed into the neural network to be recognized as number or alphabet characters.

For each license plate source image, there will be 3 to 9 characters depending on the plate. These characters are saved in the local standard SQLite database in the android-based mobile device. Information saved in the database is type of automobile (car or motorcycle), license plate characters, and the time of record. Receipt of the transaction record can be printed using Bluetooth or Wifi interface to a remote printer.

V. Results and Discussion

Even though the training result shows that the accuracy is as high as 88.2%, it is in the contrary with the test result that was performed in android-based mobile device. The result of image recognition accuracy that is tested is shown in table 1 where all license plates that were captured were in good condition. It is seen that although the same license plate was captured two times, but the reading result is different from one image to another. Lightning position as well as camera position played important role with the image recognition that should be controlled since the beginning of capturing the image.

In the table 2, we did experiment by capturing the image of license plate directly from the android-based mobile device (dual core 1GHz) where some of the images were not well-captured. It needed 15 to 20 seconds for the device to recognize each image and it is seen that many of the recognitions are failed. The image segmentation could not be performed properly due to the poor images that were manually captured by mobile device. This poor performance also found when the license plate was not in the good condition such as there was smooth crack in between the letters or pointing bolts under the letters.

After image recognition process was done, the reading then saved in database and in the form of file.txt format. The database then is used as parking report which is contained of type of vehicle, license plate number, and time. The file.txt will be printed out by using PrintShare application and can be used as parking receipt for the driver or the owner of the vehicle.

VI. Conclusion

This simple experiment is done in order to build a license place recognition application running on mobile device with android operating system, which is an open source operating system. Even though the result is still far from what is expected for practical use, this small step can enhance advanced development of this application in the near future for road side automobile transaction system, such as parking and ticketing.

TABLE 1
TEST RESULTS FOR LICENSE PLATE RECOGNITION IN GOOD CONDITION

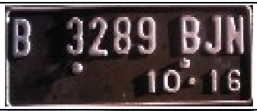
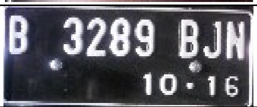
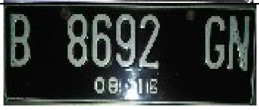
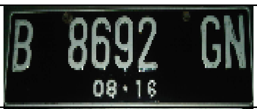
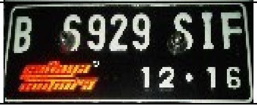
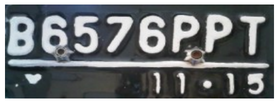
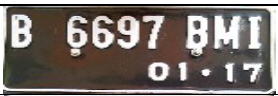
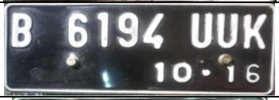
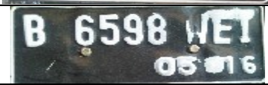
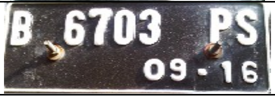
No	Image Capture	Recognition Result	Recognition Percentage
1		B3289BJH	87.5 %
2		D328SHJH	50 %
3		B8692LH	71.43 %
4		B85926N	71.43 %
05		HS929SIF	75 %
		Average:	71%

TABLE 2
TEST RESULTS FOR LICENSE PLATE RECOGNITION WHEN TAKING DIRECTLY FROM ANDROID-BASED DEVICE

No	Image Capture	Recognition Result	Recognition Percentage
1		J157GPPT	63%
2		H66S7OKI	50%
3		H81S4DCK	38%
4		B6598I	75%
5		B6J08D9	43%
		Average:	54%

References

[1] Weihua Wang. *License Plate Recognition Algorithm Based on Radial Basis Function Neural Networks*. Proceeding of International Symposium on Intelligent Ubiquitous Computing and Education; 2009, p. 38-41.

[2] Ning Chen, Li Xing. *Research of License Plate Recognition based on Improved BP Neural Network*. Proceeding of International Conference on Computer Application and System Modeling; 2010, p. V11-482 - V11-485.

[3] Hsin-Fu Chen, Chang-Yun Chiang, Shih-Jui Yang, Ho, C.C. *Android-Based Patrol Robot Featuring Automatic License Plate Recognition*.

Proceeding of Computing, Communications and Applications Conference; 2012, p. 117-122.

[4] Mutholib, A.; Gunawan, T.S.; Kartiwi, M. *Design and implementation of automatic number plate recognition on android platform*. Proceeding of International Conference on Computer and Communication Engineering; 2012, p. 540-543.

[5] Zheng L., Samali B., He X., and Yang L.T. *Accuracy Enhancement for License Plate Recognition*. Proceeding of 10th IEEE International Conference on Computer and Information Technology (CIT); 2010, p. 511-516.

[6] Kasaei, S.H.M., and Kasaei S.M.M. *Extractation and Recognition of the Vehicle License Plate for Passing Under Outside Environment*. Proceeding of European Intelligence and Security Informatics Conference; 2011, p. 234-237.

[7] Megalingam R.K., Krishna P., Somarajan P., Pillai V.A., and Hakkim R.U.I. *Extraction of License Plate Region in Automatic License Plate Recognition*. Proceeding of International Conference on Mechanical and Electrical Technology (ICMET); 2010, p. 496-501.

[8] Feng Z., and Fang K. *Research and Implementation of an Improved License Plate Recognition Algorithm.* Proceeding of 4th International Conference on Biomedical Engineering and Informatics (BMEI); 2011, p. 2300-2305.

[9] Vesnin E.N. and Tsarev V.A. *Segmentation of Images of License Plates*. Pattern Recognition and Image Analysis **16** (2006) 1, pp. 108–110.

[10] Sirithinaphong T. and Chamnongthai K. *The Recognition of Car License Plate for Automatic Parking System*. Proceeding of 5th International Symposium on Signal Processing and Its Application, Brisbane; 1999, p. 455-457.

[11] Beale M.H., Hagan M.T., and Demuth H.B. *Neural Network Toolbox^{TM} – User's Guide*. The MathWorks, Inc.; 2012.

[12] Greensted A. *Otsu Thresholding.* Retrieved on Nov 1st, 2011, from The Lab Book Pages: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html.

[13] Anonymous. *Global Image Threshold Using Otsu's Method – MATLAB*. Retrieved on Nov 1st, 2011, from MathWorks: http://www.mathworks.com / help/toolbox/images/ref/graythresh.html.