License Plate Detection and Recognition System

Harsimran Kaur, Ruchika, Gurleen Kaur and Paras Verma

***Abstract*—Automatic License Plate Detection and Recognition (ALPR) systems have become important for the security and the safety of cities and for controlling traffic congestion problems. Various methodologies to implement the system have been examined in the paper. All methodologies are tested over the same dataset created using images from the internet and Kaggle. The test results are analyzed for five categories of images : (1) ideal image (2) noisy image (3) blurred image (4) rotated image (5) black and white image. Based on the test results, the methodologies that could undergo re-training on data were trained and analysed again. Moreover, an attempt is made to propose an optimal solution formed from the combination of all the methodologies explained to efficiently detect and recognize license plates images of all the five categories.**

***Keywords*—License Plate Detection, Character Recognition, Image Processing, Machine Learning, Kaggle, Tesseract**

I. INTRODUCTION

Automatic Licence Plate Recognition System (ALPR) has found its way in the research area owing to its varied applications, such as the payment of parking fees, highway toll fees, traffic data collection, and crime prevention. The underlying framework of these algorithms has basic tasks in common like image acquisition, image pre-processing, plate detection, and character processing . The four algorithms studied are:

1) License plate detection and recognition using Connected Component Analysis (CCA) and Neural Networks[2].
2) License Plate Detection and Recognition using OpenCV and K-Nearest Neighbour Classifier(KNN)[4].
3) Automatic License Plate Detection and Recognition using Deep Learning[10].
4) Car Number Plate Detection Using MATLAB and Image Processing[9].

Some of the algorithms are based on new techniques of machine learning and computer vision that use OpenCV while other algorithms use MATLAB image processing, relatively not new in domain . Some of the algorithms perform plate detection and then character recognition but the algorithm that uses MATLAB image processing only detects the characters. All the algorithms are analysed on the basis of their results, advantages and limitations in section II

II. METHODOLOGIES

Some of the various techniques being practiced worldwide are discussed below.

A. License plate detection and recognition using CCA[I] and Neural Networks

The approach used in this algorithm[2] has four main steps: (1) Detection of License Plate (2) Character Segmentation (3) Training of Machine Learning Model (4) Prediction of characters in License Plate.

In the Detection step, CCA[I] has been used to identify the pixels that belong to the same object. A car image is passed to the system and is first converted into a grayscale image. The image is further converted into a binary image after applying a threshold using the scikit-image library. CCA[I] is performed on the binary image. Various connected regions are identified and a series of operations are performed to satisfy the condition of a typical license plate on the regions. A license plate is identified and extracted from the regions. The extracted plate image is inverted from black pixels to white pixels and CCA[I] is done to identify various regions that represent characters. Then, a red rectangular box is plotted around those regions that satisfy the conditions of a character to segment the characters. A training dataset of characters(A-B and 0-9) is used to train a model. SVC (4-cross-fold validation) is used for training and the model is loaded. The segmentation result is passed to the model to predict the characters. In the original system, a custom dataset of images of various

characters is used. However, it limits to only 10 different images per character due to which the system doesn't recognize some characters of a license plate. So, we tried to train this system with the chars74k[5] character dataset. On testing, the accuracy increased from 61% to 79%.

B. License Plate Detection and Recognition using OpenCV and KNN[I] Classifier.

The algorithm[4] has two main parts: (1) Detection and (2) KNN[I]-Character Recognition. The algorithm has two main parts: (1) Detection and (2) KNN[I]-Character Recognition[20]. The Detection algorithm uses OpenCV functionalities to identify the license plate in the input image or video frame. The input is pre-processed using Gaussian Blur[16] and Adaptive thresholding[15] to remove noise. Initially, a rough guess is made for the area of contour of each character to find all the possible characters in the image. The license plates are detected by forming bounding boxes to fit characters that occur together on the processed image. The KNN[I] Character Recognition algorithm uses a training image of characters (alphabets(A-Z) and digits(0-9)) to recognize characters. The contours found in the training image are iterated while pressing their respective keys from the keyboard as indicated by the Region of Interest(ROI)[21]. This leads to the formation of two files: classification.txt and FlattenImage.txt. The "Classification" file contains the integral value of a character corresponding to its image represented as multiple 1-Dimension arrays in the "FlattenImage" file. The classifier is trained over each label("classification" file values) and image("FlattenImage" file values) pair.

The trained KNN[I] classifier is then used to identify characters on each possible plate detected. A list of strings is formed by the outputs of all the plate images and the maximum length string corresponds to the license plate detected in the image. The classifier can be trained over various fonts of each character as all characters are identified by its ASCII value from the keyboard input. This enhances the accuracy of the recognition algorithm from 82% to 89%. However, the classifier cannot be trained on characters that are not available on the keyboard. Moreover, the rough guess may produce false output for zoomed images as the values are hard-coded.

C. Automatic License Plate Detection and Recognition using Deep Learning.

The algorithm[10] follows three main steps:(1) Detection (2) Segmentation (3) Recognition. To detect the license plate YOLO(You Only Look Once)[11], a convolution neural network architecture, is used because it gives high accuracy and speed(45 frames per second).[6] The YOLO model is retrained over a dataset composed of 700 images of cars that contain the Tunisian license plate which is labeled using "LabelImg", a graphical annotation tool[11]. Then segmentation is used to extract each character on the plate detected. Summation of histogram projections is used for recognizing each character of the plates[11]. A pre-trained Convolutional Neural Network(CNN) model trained on the MNIST dataset[13] is retrained over a dataset of Tunisian words in Arabic and digits(0-9)[1]. The Multilayer perceptron (MLP)[11] model created is used for the recognition step. The algorithms perform well on Tunisian plates with an accuracy of 91% for detection and 89% for recognition. The hidden layers of the model and the YOLO weights used in the process increases the accuracy of detecting and recognizing the license plate. To make the algorithm respond to multi-state plates, we trained the YOLO model over a dataset composed of 700 license plate images of each state. However, plates are not identified if the image is small in size and if contains more than one plate in an image. To enhance the recognition accuracy, we tried to train the recognition model on multi-language words and multiple-fonts of each word[11]. This lead to a 4% increase in accuracy since the model used a pre-trained CNN model which already performed well for various fonts.

D. Car Number Plate Detection Using MATLAB and Image Processing

The algorithm[9] recognizes the characters by comparing the input image of a license plate with different plate templates provided in the alpha folder by using MATLAB image processing. The number is returned in text format. This approach works well only for high-resolution images giving 84% accuracy. First, the input image is resized for comparison with the templates provided and converted to grayscale. The grayscale image is converted to

binary to detect edges using the Prewitt method. Then the image is compared with character templates and the correlation of the input image with every image in the template takes place to get the best match. The index is found that corresponds to the highest matched character and that respective character is printed. More alphanumeric images from the scikit dataset[14] were added in the alpha folder by modifying 'template_creation.m', keeping the size of templates the same for comparison to increase recognition accuracy by 3%. To further improve the recognition method, the variations in the background were reduced using 'imbinarize' function. The digits in the template folder can be replaced with the MNIST dataset[13] to detect handwritten numbers. The algorithms involve no training. Thus, it is easily modifiable and scalable.

III. TEST DATA FORMATION

All the methodologies explained in section II were trained on different and versatile types of training data. So, the idea of training them over the same data was incompetent owing to the limitations of each methodology. Therefore, the optimal way of analyzing and comparing the methodologies was to test them over the same test data and store their output against it.

The test data was enriched with variety of images of cars with visible license plates. Versatility was introduced to the data by adding multiple states and languages license plate images.Images were downloaded from sources like Google, Yahoo, Bing, etc. Kaggle links [17][12][8] resulted to be a trump card for the formation of test data.

Once the images were collected, data augmentation was used to enhance the dataset. Data Augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models without actually collecting new data[3]. Keras's image data generator [7] was used to augment the data on various factors like noise, blur, color, rotation, shift, zoom, padding, flipping, etc. The data were classified in groups (sub-folders) based on the type of images like ideal image, noisy and blurred image, rotated image, black and white image with at least 100 images per group. The data formed was tested on all the methodologies and the results were analyzed based on the accuracy of detection and recognition of the license plates in various groups.

IV. TESTING ANALYSIS

After testing the methodologies in section II with the data, it was analyzed that all the algorithms detected and recognised plates perfectly with an average accuracy of 86% among all the algorithms if the image is clear and has high resolution as depicted in figure 1

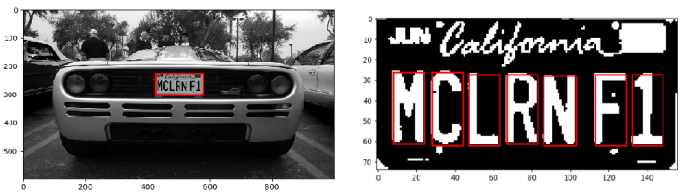

Fig. 1. Output of Methdology [II-A]

However, several variations were seen in the other categories of images when tested all the algorithms as depicted in figure 2

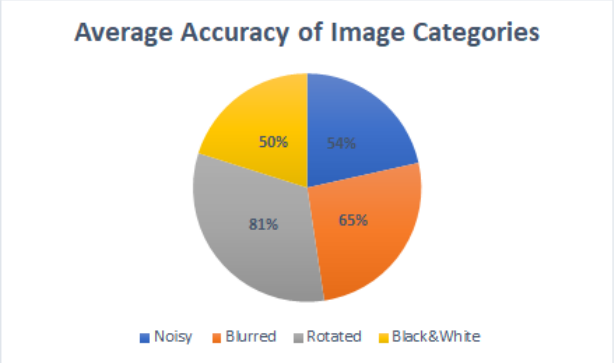

Fig. 2. Average accuracy of Image Categories

It was concluded that license plates were detected and recognised the best when the image was rotated as compared to noisy, blurred or black and white image transformations. This conclusion led to the fact that the algorithms will work with sufficiently good accuracy even when the image is taken from an angle. Black and white image category was formed so that the cost of the hardware, if normal CCTV cameras are used, could be minimised. However, the results were poor on black

and white images, which could led to removal of the grayscale conversion, a pre-processing step, to test its accuracy. This change increased the black and white accuracy by approx 32%.

It is possible to re-train the algorithm II-C on new images and add new templates in the algorithm II-D. So, the 70% of this test data was used to enhance both the algorithms. The rest 30% data was used to further test the updated algorithms. An increase in the accuracy of the algorithms was observed as depicted by figure 3

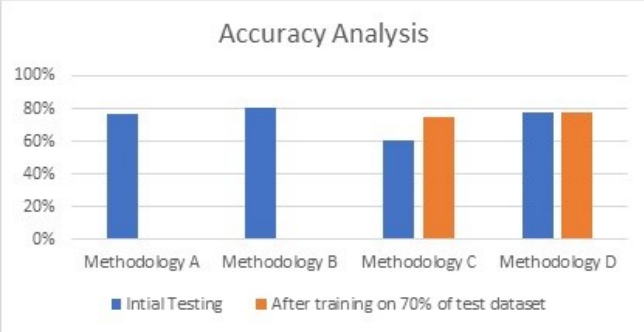

Fig. 3. Accuracy Analysis

Therefore, it can be concluded that the algorithms that required templates or a model to detect and recognise plates worked with sufficiently good accuracy only if they are trained with accurate data. However, the algorithms II-A and II-B, worked well despite of any prior training and worked as a more general solution as compared to the other algorithms and are the recommended one when specification to a particular state/language is not necessary. Owing to their respective limitations described in II-A and II-B, a further optical solution is proposed in the next section.

V. OPTIMAL SOLUTION

After the analysis on the test data created, it was observed that all the methodologies II mostly used Machine/Deep Learning to detect and recognize license plate numbers. However, Machine Learning/Deep Learning approach highly depends on the availability of large amounts of relevant data. An optimal solution of license plate detection can be achieved through Image Processing. The solution is proposed to minimize the effort and cost in creating gathering relevant data for the Machine Learning approaches. The solution pre-processes the image using grayscale conversion, Adaptive Mean and Adaptive Gaussian thresholding to easily identify and recognize the characters on the plates. All the rectangles present in the processed image are identified and assumed to be plates. Tesseract is an optical character recognition engine for various operating systems. It is considered one of the most accurate open-source OCR engines available[22]. All plates are cropped and Tesseract wrapper in python (Pytesseract) is used to identify characters in the cropped image using its simple function “image_to_string()”. The string returned by the function is further verified against a list of possible characters. Tesseract has multi-language support[18]. Therefore, it can be efficiently used to identify any state’s license plates. This solution was tested and analyzed on the dataset created in section III and resulted in 13% higher accuracy as compared to the other methodologies in section II.

VI. CONCLUSION

Since the original algorithms were trained over the dataset-specific to countries, training algorithm II-C on a new dataset that was created by merging multiple datasets available on “Kaggle”, proved to increase their efficiency many folds. The algorithm II-D that involved no training of data were introduced with new templates to recognize license plate characters. Moreover, smoothening filters like grayscale conversion and Gaussian filters were altered to enhance efficiency. Apart from introducing modifications to algorithms and testing over the same dataset, an optimal solutionV suggested a general approach to detect and recognize the license plates. The solution used Tesseract output verified against a list of possible characters. To further enhance the efficiency, Tesseract provides easy multiple ways of retraining its model to include variety in the dataset[19]. Thus, it proved to be a great source for usage in detecting and recognizing a license plate.

REFERENCES

[1] M. Loey A. Elsawy, H.M.E. Bakry. Arabic Handwritten Characters Recognition using Convolutional Neural Network. https://www.researchgate.net/publication/313891953_Arabic_Handwritten_Characters_Recognition_using_Convolutional_Neural_Network, 2019. [Online; accessed 11-November-2019].

[2] W. I. Rahim A.Dave. License Plate Detector. https://github.com/apoorva-dave/LicensePlateDetectorl, 2018-2019. [Online; accessed 11-November-2019].

[3] R. Liaw D. Ho, E. Liang. 1000x Faster Data Augmentation. https://bair.berkeley.edu/blog/2019/06/07/data_aug/, 2019. [Online; accessed 14-November-2019].

[4] C. Dahms. OpenCV3 License Plate Recognition Python. https://github.com/MicrocontrollersAndMore/OpenCV_3_License_Plate_Recognition_Python, 2017. [Online; accessed 11-November-2019].

[5] T. de Campos. Chars74K Dataset. http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/l, 2012. [Online; accessed 15-november-2019].

[6] M. David Fudenberg, Drew & Kreps. You Only Look Once: Unified, Real-Time Object Detection. 2016. [Online; accessed 13-November-2019].

[7] Keras. Image Processing. https://keras.io/preprocessing/image/, 2019. [Online; accessed 14-November-2019].

[8] U. Khan. Car Number Plate. https://www.kaggle.com/mrugankray/license-plates, 2017. [Online; accessed 10-November-2019].

[9] P. Khatri. Car Number Plate Detection Using MATLAB and Image Processing. https://circuitdigest.com/tutorial/vehicle-number-plate-detection-using-matlab-and-image-processing, 2018. [Online; accessed 9-November-2019].

[10] A. Khazri. Automatic License plate Detection Recognition using deep learning. https://github.com/GuiltyNeuron/ANPR, 2019. [Online; accessed 11-November-2019].

[11] A. Khazri. Automatic License plate Detection Recognition using deep learning. https://towardsdatascience.com/automatic-license-plate_-detection-recognition-using-deep-learning-624def07eaaf, 2019. [Online; accessed 11-November-2019].

[12] M. Kray. License Plates. https://www.kaggle.com/mrugankray/license-plates, 2018. [Online; accessed 10-November-2019].

[13] Corinna & Burges CJ LeCun, Yann & Cortes. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[14] Scikit-Learn Developers (BSD License). Scikit Dataset. https://scikit-learn.org/stable/datasets/index.html, 2007-2008. [Online; accessed 11-November-2019].

[15] Scikit-Learn Developers (BSD License). Scikit-Image. https://scikit-image.org/docs/0.12.x/auto_examples/segmentation/plot_threshold_adaptive, 2019. [Online; accessed 13-November-2019].

[16] A. Walker E. Wolfart R. Fisher, S. Perkins. Gaussian Blur. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm, 2003. [Online; accessed 13-November-2019].

[17] T. Selvan. Indian License Plates. https://www.kaggle.com/thamizhsterio/indian-license-plates, 2018. [Online; accessed 10-November-2019].

[18] Tesseract-OCR. Tessearct. https://github.com/tesseract-ocr/tesseract, 2019. [Online; accessed 16-November-2019].

[19] Tesseract-OCR. Tessearct. https://github.com/tesseract-ocr/tesseract/wiki/TrainingTesseract, 2019. [Online; accessed 16-November-2019].

[20] Wikipedia. K-Nearest Neighbour Algorithm . https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm, 2019. [Online; accessed 12-November-2019].

[21] Wikipedia. Region of interest. https://en.wikipedia.org/wiki/Region_of_interest, 2019. [Online; accessed 11-November-2019].

[22] Wikipedia. Tessearct. https://en.wikipedia.org/wiki/Tesseract, 2019. [Online; accessed 16-November-2019].