

Multi-String Missing Characters Restoration for Automatic License Plate Recognition System

Ishtiaq Rasool KHAN¹, Syed Talha Abid ALI², Asif SIDDIQ³, Seong-O SHIM⁴

College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia^{1,4}
Dept. Electrical Engineering, Pakistan Institute of Engineering and Technology, Multan, Pakistan^{2,3}

Abstract—Developing a license plate recognition system that can cope with unconstrained real-time scenarios is very challenging. Additional cues, such as the color and dimensions of the plate, and font of the text, can be useful in improving the system's accuracy. This paper presents a deep learning-based plate recognition system that can take advantage of the bilingual text in the license plates, as used in many countries, including Saudi Arabia. We train and test the model using a custom dataset generated from real-time traffic videos in Saudi Arabia. Using the English alphanumeric alone, the accuracy of our system was on par with the existing state-of-the-art algorithms. However, it increased significantly when the additional information from the detection of Arabic text was utilized. We propose a new algorithm to restore noise-affected missing or misidentified characters in the plate. We generated a new test dataset of license plates to test how the proposed system performs in challenging scenarios. The results show a clear advantage of the proposed system over several commercially available solutions, including Open ALPR, Plate Recognizer, and Sighthound.

Keywords—Automatic License Plate Recognition (ALPR); Intelligent Transportation System (ITS); Optical Character Recognition (OCR); Deep Convolutional Neural Network; You Only Look Once (YOLO)

I. INTRODUCTION

In recent times, smart cities have been a popular trend, resulting in the pacing of the development of several enabling technologies. One of them is robust automatic license plate recognition (ALPR) [1], which offers a computer vision based solution for intelligent transportation systems (ITS) [2]. In this regard, a passive mesh of cameras is generally installed at road intersections and other suitable locations to observe vehicle routing through urban environments [3]. ITS can improve safety and mobility and help law enforcement agencies monitor traffic effectively. There have been several systems proposed to detect and recognize license plates (LPs) for various applications like toll fees collection [4], monitoring for the speed of a car on the road [5], traffic volume estimation on [6], detection of illegal parking [7], highway surveillance [8], and border control [9]. However, most systems tackling any of these aspects often work well only when operated in a restricted environment where camera distance and angle are fixed [10]. In an unconstrained environment, factors like low image resolution, dynamic background, motion blur [11], and variable lighting conditions, can degrade the image quality. This makes recognizing LP characters in unconstrained environments a very challenging task.

Some solutions that have been proposed to address these variabilities include Laplacian gradient-based partial character segmentation [12], Generative Adversarial Network (GAN) [13], super-resolution (SR) image reconstruction using maximum a posteriori (MAP) [14], SR reconstruction [15], and Recurrent Neural Network (RNN) [16]. Although these approaches can help restore the deteriorated or missed characters within LP, they do it at the expense of higher computational complexity due to the additional machine learning model used for the restoration task. Similarly, methods like multiscale adaptive thresholding [17] and matrix edge information based adaptive thresholding [18] depend on edge information. They can produce false positives due to weak edges, especially in the presence of high noise.

We propose a simple and fast technique for detecting and recognizing the contents of LP in unconstrained environments. You Only Look Once Version 5 (YOLOv5) network is used for detection, while a custom-built convolutional neural network (CNN) recognizes the contents in the detected LP. We collected a custom dataset from real-time traffic in Saudi Arabia, where the number plates are bilingual, using Arabic text on top and English at the bottom. The unique feature of the proposed method compared to the existing works is that it tries to recognize alphanumerics in both languages and then combines both results to improve the overall accuracy of LP recognition. We propose a new algorithm, which can address several scenarios of unrecognized and wrongly recognized alphanumerics. It uses computationally simple and efficient techniques, implemented without using a separate deep learning model, and does not depend on the deteriorated character's edge information either to restore the noise-affected missed character. Our results show that the proposed model outperforms commercially available server-based ALPR systems of Sighthound [19], Plate Recognizer [20], and Open ALPR [21].

The rest of the paper is organized as follows. Section II gives an overview of the related work. Section III describes training dataset preparation steps, including extracting LP from the captured video streams and synthetically generating and augmenting more plates to add to the data. The steps of detecting LP, recognizing Arabic and English alphanumeric characters, and restoration of missed and wrongly recognized characters are discussed in Section IV. Experimental evaluation is reported in Section V, while some conclusions and future research directions are given in Section VI.

II. RELATED WORK

Over the years, numerous designs based on different architecture and technologies have been proposed to solve the detection and recognition problems of ALPR. For the detection of LPs, some works use image processing-based solutions. Chen and Luo [22] proposed an LP localization method using an improved version of the Prewitt arithmetic operator. It extracts the exact location from vertical and horizontal projections. However, the method requires prior knowledge of all the characters' textures which become difficult to obtain while working in an unconstrained real-time environment. Vijeta et al. [12] used stroke pair candidate detection by using Laplacian and gradient information. They find pixels that represent the stroke width of characters on a given LP. However, this method usually fails while dealing with complex backgrounds in real time due to unsymmetrical features within the captured frames.

YOLO models have been used to detect LPs in more recent works. Alghyaline [11] used a modified YOLOv3 model. Since the LP size is much smaller than the captured frame, they used only 15 layers instead of the original 75. It made the process fast but at the expense of low accuracy when detecting smaller characters within LPs. Ju-Yeong Sung [23] used YOLOv4, which can detect smaller objects, both LP and the characters within it, but at the expense of higher training costs. Khan et al. [24] used the YOLOv5 model to detect LP and its characters in a real-time environment. The YOLOv5 model uses path aggregators to speed up the detection process in real time. It also uses the mosaic data augmentation technique, which helps detect very small objects in challenging environments. Moreover, it also uses auto-learning bounding box anchors, which help make bounding boxes around objects despite the challenging backgrounds within a frame.

Some works use image processing-based techniques to reconstruct deteriorated or missed characters for the recognition task. Vijeta [12] proposed partial character reconstruction with the Tesseract optical character recognition (OCR) library, which uses the characteristics of stroke widths in the Laplacian and gradient domains. The method first enhances high-contrast information at the edges using symmetrical features of incomplete characters by suppressing background information. Then it uses stroke width properties to reconstruct the complete shape of the deteriorated characters. This method, however, proves to be not very effective for restoring completely deteriorated characters. Khoshki et al. [17] proposed a multiscale adaptive thresholding method for ALPR, which is used to find the candidates matching the LP characters affected by noise. However, the effectivity of this method lessens under varying illumination conditions. Moreover, it is computationally intensive and requires significant time to process input images. Mokji et al. [18] proposed an algorithm that incorporates matrix edge information, which enhances foreground objects in relation to the neighborhood pixels. This method does not work well when the edges of objects are weak and disconnected. Moreover, in an extremely degraded image, strong interfering patterns can generate false edges.

Some works used SR algorithms along with machine learning models. Lin et al. [13] proposed an SR image reconstruction method using GAN, an unsupervised learning model. It preprocesses the affected input image and extracts image features using a residual dense network. The method then uses sampling to restore more information using a larger receptive field. A Markovian discriminator is used to accurately guide the generator to reconstruct high-quality restored images. Despite good results, the method increases the difficulty level of training such a highly complex model. Moreover, it shows edge artifacts as well. Zhan et al. [14] proposed a MAP-based SR image reconstruction approach. It helps to estimate distribution and model parameters that best explain an observed dataset. It then uses a Huber Markov random field (HMRF) along with an ALPR system to measure image smoothness. The technique improves the recognition rate by restoring and improving the quality of LP image. However, quality enhancement is limited to single-frame reconstruction instead of a sequence of video frames.

Chen et al. [15] proposed an SR reconstruction algorithm. It uses the mechanism of attention along with a feature map to reconstruct multiscale SR images from original low-resolution images. It uses a combined feature map of multiple channels before applying a reconstruction module. It also uses interdependence to adaptively adjust the characteristics of the channel to restore details before generating high-resolution images at different scales. Despite reconstructing images, it increases weight parameters which eventually increases training time. Hui Li et al. [16] extracted and recognized the sequential features from the whole LP using a recurrent neural network with long short-term memory. However, a large number of labeled LPs are required for training. Duan et al. [25] used inception structure, which utilizes computational resources effectively by image dimensionality reduction, in an end-to-end CNN. However, this can affect the recognition accuracy of noisy images taken in real-time scenarios. Ergun et al. [26] used a statistical method in which refined characters are stretched/reduced to a given size and matched in a labeled database. This method has a limitation as it requires exact correspondence. Moreover, a slight deterioration of characters can affect the recognition accuracy.

There are several commercial ALPR solutions also available. OpenALPR [21], PlateRecognizer [20], and Sighthound[19] are open-source systems that can recognize LPs in a given input frame. These systems can be accessed using their cloud API services that predominantly use OpenCV and Tesseract OCR libraries. Moreover, reconstruction algorithms are also part of these available systems, but they require incomplete strokes of digits or characters in a noiseless background for good performance. Otherwise, these APIs fail to restore the completely missing alphanumeric in noise effected license plates.

III. THE PROPOSED METHOD

The proposed system uses several steps, including extraction of frames from the videos of traffic, annotating the frames, augmenting the dataset, training the network, and testing its performance. The complete pipeline is shown in Fig. 1, and its different modules are explained in this section.

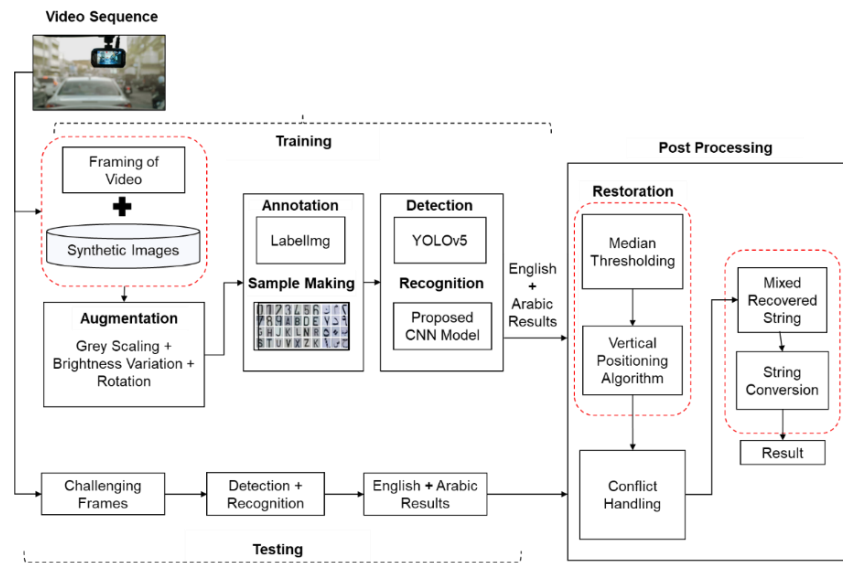


Fig. 1. The pipeline of proposed multi-string ALPR system.

A. Preparation of Dataset

1) *Data acquisition*: Designs of the most commonly used LPs in Saudi Arabia are shown in Fig. 2. The standard size of these plates is 32 cm in width and 16 cm in height. The color schemes indicate the usage of vehicles. We captured many videos of live traffic, with a sufficient number of samples of each type. We examined each frame of these videos manually and extracted 2600 frames in total. The rest of the frames that had no vehicle or were very similar to one of the previously selected frames were discarded. The frames were stored as color images in PNG format at FHD resolution (1920×1080 pixels).

2) *Synthetic Saudi Arabian LP Generation*: No Saudi LP dataset is available in the literature with sufficient size (number of images) and variety of scenarios to support robust training of LP detection and recognition models. Moreover, dataset preparation is a very time-consuming process. Hence, we increased the size of our dataset by adding some synthetically generated license plates. For this, our algorithm first generates random but mutually mapping strings of officially used English and Arabic alphanumerics. These random texts are then appended on synthetically generated layouts. The synthetic layouts contain immutable elements such as the country name, the grid, and other official symbols on the right side of every plate. The mapping coordinates are determined based on the positional analysis of different characters in the actual plates. A total of 200 synthetic images were generated this way and included in the dataset. Some examples are shown in Fig. 3. In a real-world scenario, the license plates may be dirty and less readable than the synthetically generated plates. Thus, to create more realistic data, noise is added. Afterward, all these plates are appended on some selective frames of actual data, as shown in Fig. 4.



Fig. 2. Types of Saudi Arabian license plates [27].



Fig. 3. Examples of synthetically generated Saudi Arabian license plates.



Fig. 4. Examples of appending synthetically generated plates in different video frames.

B. Training Samples Generation

All Saudi Arabian LPs have three alphabet and four digits in English and Arabic both. For training of the custom CNN model, samples for each English and Arabic Alphanumeric glyph are required. For this, an algorithm was designed to automate certain steps of this process, avoiding human intervention. YOLOv5 [28] model initially trained on the available custom dataset is used to detect LP in each frame along with all Arabic and English alphanumerics in it. Coordinates of the detected bounding boxes are used to extract the alphanumerics from the LP automatically. As the training of the network is improved, more samples can be extracted more accurately with minimal manual supervision. The process is depicted in Fig. 5.

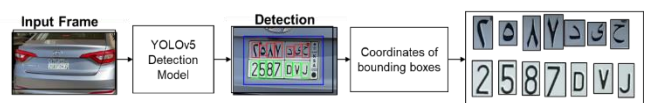


Fig. 5. Generation of training samples.

Each extracted sample is saved in its respective folder. There are 54 classes in total, including ten numerical digits (0-9) and 17 alphabets among (A-Z) that are officially used in LPs, and another 27 Arabic counterparts, containing ten digits (٠ - ٩) and 17 alphabets among (ا - ح).

C. Data Augmentation

Deep learning models require large datasets for increased accuracy. In comparison, our proprietary data is relatively limited, having 2800 frames in total, including 2600 frames extracted from the actual traffic videos and 200 generated synthetically. We split them into training and testing sets containing 2600 and 200 frames, respectively. The testing set includes challenging situations of deteriorated and completely missed alphanumeric characters in Arabic and English strings.

To increase the training dataset and its variability, data augmentation techniques were used, including gray scaling, brightness variation, and rotation. As a result, we got 6500 frames and 15000 alphanumeric characters in total. They vary considerably in size and cannot be used for training without preprocessing. We scale each of them 128x128 pixels. Through augmentation, we could increase the variability in the dataset, which helps in better training of networks, improving their accuracies, and avoiding overfitting. The numbers of frames and samples are shown in Table I.

TABLE I. SUMMARY OF THE TRAINING DATASET

Dataset	Real	Real + Augmented
Frames	2600	6500
Arabic + English Characters	9000	15000

D. Data Annotation

We used a graphical image annotation tool, LabelImg [29], to label the dataset. The coordinates of plates as well as the individual characters, are marked, as shown in Fig. 6. Since we are using YOLOv5 for detection, which accepts annotations in extensible markup language (XML) files, we save the annotations in this format.

E. Detection of License Plates

Due to their high efficiency, the detectors in the YOLO family are often used to work in real-time scenarios [30] [31] [32]. Since the release of its first version, many updates and new versions of YOLO have been developed. The most recent is YOLOv5 [28] [33], which outperforms its predecessors in terms of computational complexity and accuracy of detection. The structure of the YOLOv5 model, a relatively new family member, is shown in Fig. 7.

YOLOv5 is pre-trained on a large dataset; therefore, we use transfer learning [34] using our proprietary dataset. Transfer learning is a well-known technique to fine-tune a previously trained neural network to perform a new task. In our case, YOLOv5 is trained to perform general object detection, and we tune it for LPs. This process prevents the need to complete network training from scratch, which would have required an extensive dataset. For training YOLOv5, our training dataset is split at a ratio of 70 to 30 as shown in Table II.

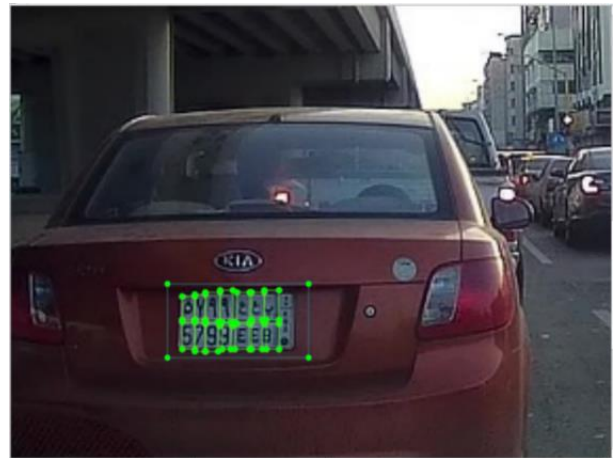


Fig. 6. Annotation of the LP and Arabic and English alphanumeric in an LP using LabelImg software tool.

TABLE II. DATASET SPLIT FOR DETECTION MODEL

Dataset	Split Percentage	Number of Frames
Training	70%	4550
Validation	30%	1950

For the training of YOLOv5, we use Google Collab notebook [28], which provides free access to powerful GPUs. The trained model takes an input frame at the dimensions of 416x416x3 and detects LP in it. The detected LP is cropped automatically, resized to the same input dimensions of 416x416x3, and fed again to the YOLOv5 model to detect its alphanumerics. Since the original frame had an FHD resolution of 1920 x 1080 pixels, the coordinates of the bounding boxes are upscaled to find the exact locations in the original frame. Some results of the extracted LPs are shown in Fig. 8 by marking the bounding boxes with red colored rectangles.

F. Custom Network for Recognition of Arabic and English Alphanumerics

The architecture of the proposed CNN for the recognition phase is shown in Fig. 9. The input of this network is the bounding box detected by the YOLOv5 in the previous step. The proposed CNN's first layer is a convolution layer with dimensions of 224x224x3; the output of YOLOv5 is resized to match it. A maxpool layer after each convolution layer uses 2x2 pixels for pooling. The convolution layers have different filter sizes (64, 128, 256, 512), and the convolution kernel size is 3x3. There are two fully connected (FC) layers, in the end, to keep the model end-to-end trainable. "Same Padding" is used to handle the convolution near the boundaries of the image, and the stride size is 1. Adam optimizer alpha is used for its desirable characteristics in non-convex optimization problems [35].

Alternating convolutional and non-linear activation layers extract rich features of given alphanumerics. The activation function of Rectified Linear Unit (ReLU) with a stride size of 1 pixel for convolutional layers and 2 pixels for the maxpool layers is used. Each of the output FC layers contains 1000 neurons. The final decision is made by using SoftMax about the recognition (classification) of alphanumeric characters.

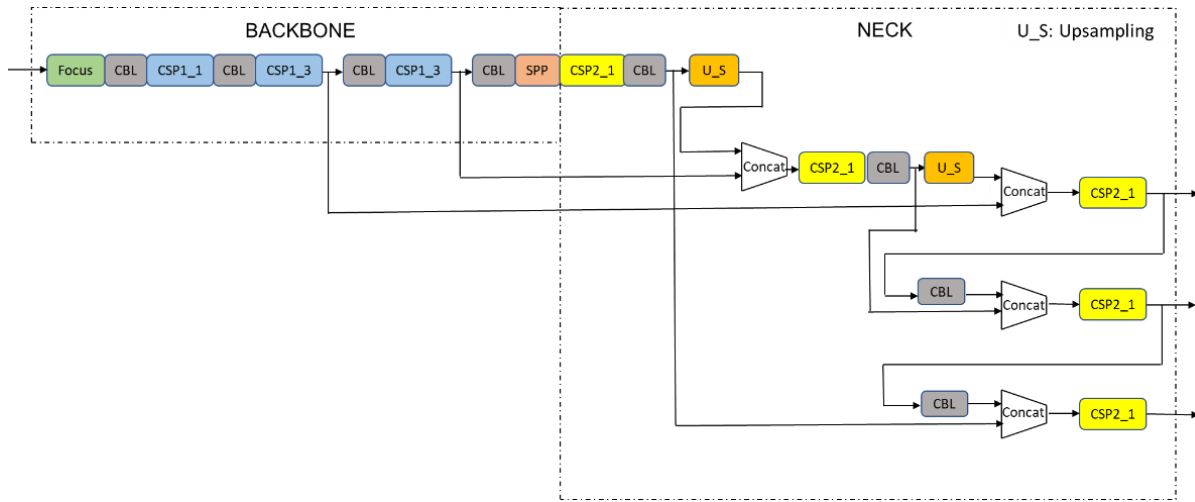


Fig. 7. CSP structure for YOLOv5 model.

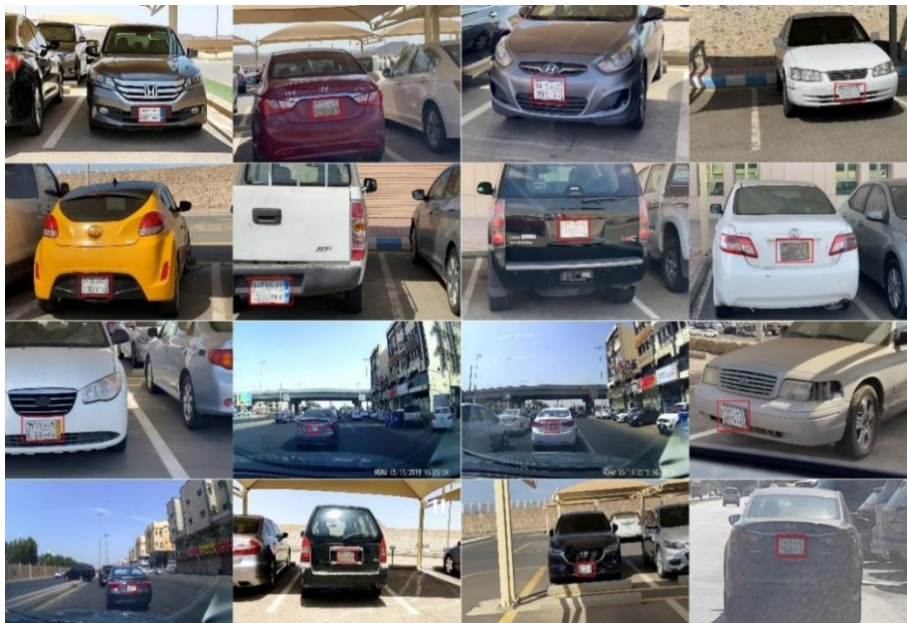


Fig. 8. YOLOv5 detection of license plates.

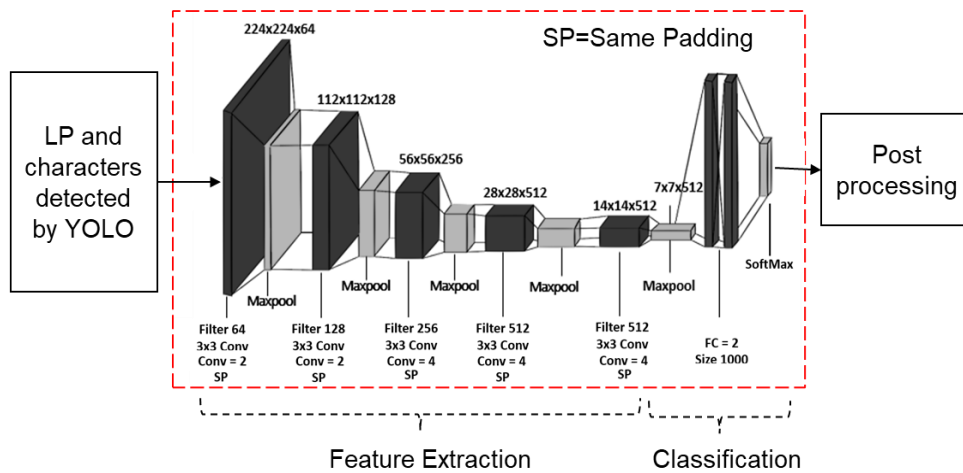


Fig. 9. The complete architecture of the proposed pipeline.

We use 15000 alphanumeric characters, including those taken from original plates and others obtained by augmentation. These are split into training, validation, and testing sets, as given in Table III.

TABLE III. DATASET SPLIT OF ARABIC AND ENGLISH ALPHANUMERIC CHARACTERS

Dataset	Split Percentage	Number of Samples
Training	70%	4550
Validation	20%	1950
Testing	10%	1500

G. Post Processing

The dataset used for testing includes some challenging scenarios. In some instances, characters are badly affected by noise, or the plate has a poor condition. Some examples are shown in Fig. 10.

The reconstruction techniques in the existing literature, as discussed earlier, can be used to restore missing and noise-affected individual characters either in English or Arabic string. However, using a separate trained restoration model will increase the whole system's computational cost. We propose a fast and simple solution based on the fact that there is redundant information available in the form of Arabic and English text, and correct recognition of the individual alphanumeric in either of them can lead to overall correct recognition of the LP.

The first step for restoration is to find the location of missing characters in Arabic and English text, and we use a median thresholding based approach for this. It measures the position of each alphanumeric bounding box in both strings relative to the extreme left of the LP and calculates the difference between the successive bounding boxes. If a character is missing, the difference between its left and right detected neighboring boxes would be large. Due to the different shapes of characters, the distance between each pair of neighbors is not the same; however, a missing character makes it much large. We calculate the median value of the distance between detected neighboring boxes and set a threshold of 1.5 times the median distance to detect the missing characters. If the distance is more than 2.5 times the median distance, we assume two characters are missing. However, to explain the algorithm, we assume that only one character is missing between two successfully detected characters. For this case, different steps involved in the restoration process are explained in Fig. 11.

The next step is to restore the missing digit or letter on the spots identified by the median thresholding algorithms above. The exact location of the missing character is assumed to be in the middle of the neighboring characters as shown in Fig. 12.



Fig. 10. Badly noise affected license plates.

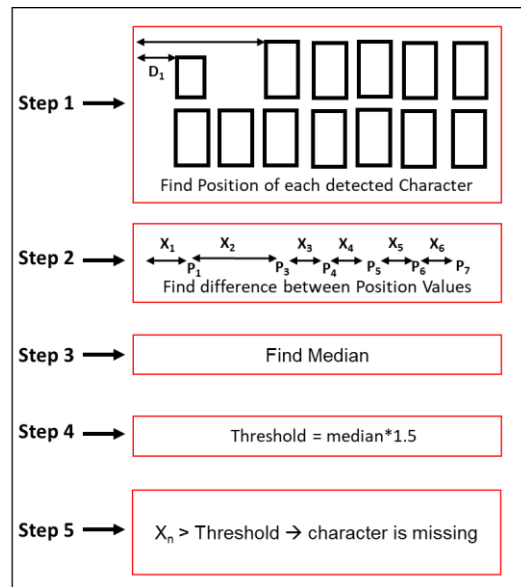


Fig. 11. Working of median threshold-based algorithm for locating missing characters.

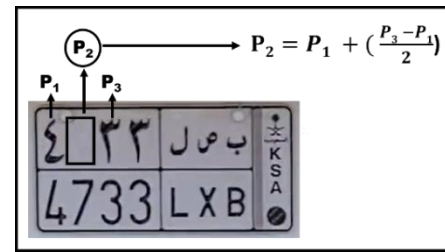


Fig. 12. Finding position of the missing character.

This way, we get seven bounding boxes in each string, regardless the character is detected or missing. The empty boxes are filled by the corresponding vertically mapping characters in the other string. Except for a rare scenario where the same character is missing in both English and Arabic strings, the algorithm can successfully recognize the plate quite accurately, as shown in the experimental results in the next section.

1) Handling conflicts between two strings: Arabic and English alphanumerics have one-to-one matches. To have recognition results with greater precision, we pick the alphanumeric, which is recognized with a higher confidence level by CNN. The string obtained this way is a mix of Arabic and English characters as shown in Table IV. We convert the entire string to the desired language, which is English in our case as shown in the table.

TABLE IV. EXAMPLES OF IMPROVED RECOGNITION RESULTS OF THE PROPOSED ALGORITHM

License Plates			
Mixed Strings	١٠٣٨TKA	١٨٠٦XHJ	21٨٢XDD
Converted Strings	1038TKA	1806XHJ	2182XDD

2) *Handling misrecognition due to bolts on the lp:* Among many challenging scenarios we encountered in our dataset, one worth mentioning is the wrong recognition of alphanumeric due to the bolts used to attach the plate to the car. There are two bolts used on the top-right and top-left regions of the plate. The location of the right bolt is such that if there is the Arabic character "Alif" (Equivalent of English character A) in the rightmost position, it is read as English "9", as shown in Fig. 13. This is, however, easy to correct, knowing that the rightmost alphanumeric cannot be a digit and it must be a character. So, if a nine is detected at this position, there is a good chance it should be "Alif". If the English string does not detect anything at the rightmost location, we take it as "Alif" or "A". However, if the English string detects something different, we include that in the final result.

Correcting the left bolt is not that straightforward, which happens to be between the first two characters in the Arabic text. If a "9" is detected as the second digit from the left, it could be actually "9" or "1". We examined several cases and found that the head of true "9" has a hollow space, whereas the head formed by the bolt is solid black. We have shown both cases side by side in Fig. 14. Our algorithm crops the head when a "9" is detected at the second place from the left and examines it further. A simple count of black pixels in the binarised image of the head can reveal if it was all filled with black pixels or had a hollow white space.

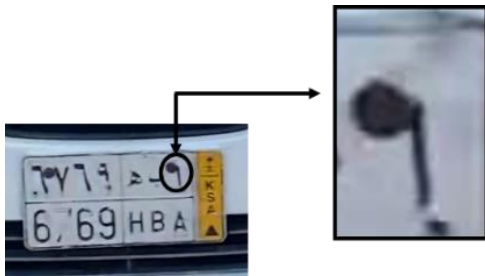


Fig. 13. Bolt Overlapping onto Arabic corresponding character of A.

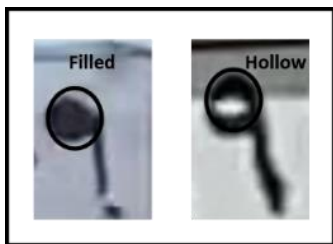


Fig. 14. One misread as nine, shown on the left, and a true 9 in Arabic shown on the right.

IV. EXPERIMENTAL EVALUATION

Training and validation are done using the official notebook repository of YOLOv5, which provides a powerful GPU for fast processing. The YOLOv5 model has 476 layers. The batch size and epoch values were set to 35 and 100, respectively. The accuracy of detection can be determined by the overlap between the annotated (ground truth) mask and detected plates. The ratio is called Intersection Over Union (IOU). We considered different values of IOU and the optimal

detection results are achieved at $IOU > 0.5$. The rectangles predicted by the YOLOv5 model below this value are discarded.

For training the proposed CNN model, a computer with moderate specifications – GPU GeForce GTX 1080 GPU, 8 GB memory – running on the Linux operating system of Ubuntu 20.04.3 LTS was used. The learning rate was set to 0.001.

The proposed system achieved significantly higher accuracy when both Arabic and English alphanumerics of the LPs were used as discussed above, compared to the accuracy of either string recognized individually. We tested a few commercially available tools as well. On good quality images, all of them achieved high accuracy. However, on challenging cases consisting of real traffic scenarios in different conditions, commercially available tools performed very poorly, and our proposed model outperformed them by a clear margin. We show some results in Tables V to VII. In the shown LP text, the red color shows a wrongly recognized character, the black color shows a correctly read character, and the blue color shows a successfully restored character.

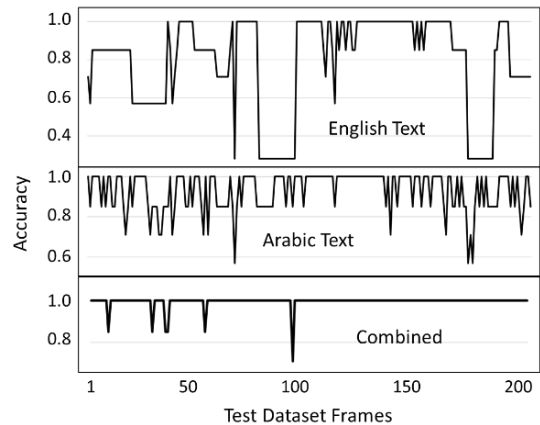


Fig. 15. Accuracy of English and Arabic strings recognition and the combined results for each LP in our test dataset.

A. Plates with Extremely Poor Visibility

This test dataset consists of cases where visibility of alphanumeric in the LP plate was poor due to shadows or glare. Many of the alphanumerics were wrongly recognized, but when we considered both resultant strings, the actual LP text was successfully restored. Some examples are shown in Table V A and V B. Note that all three commercial software failed in all these cases, whereas the proposed successfully recognized all plates.

TABLE V. A. RESULTS OF EXTREMELY POOR VISIBILITY CASES

Model				
Sighthound	7172LA	8302KEJ	3636NHB	4810VVA
OpenALPR	7172LA	6302KEJ	3636NHB	4810VA
Plate Recognizer	7172ZGJ	6302KEJ	3636NHB	4310VA
Proposed	7170DLA	5802KEJ	8636NHB	4210NVA

TABLE V. B. MORE RESULTS OF EXTREMELY POOR VISIBILITY CASES

Model			
Sighthound	5941G1	7779KXJ	3864B
OpenALPR	594LG1	7778KXJ	386B
Plate Recognizer	584LG1	7779XXJ	8664JB
Proposed	5943LGJ	7779XXJ	8864ZJB

B. Deteriorated Alphanumeric Characters

Our dataset contains some cases where alphanumerics are degraded partially or fully causing missing a few alphanumerics at the recognition stage. We show some such cases in Table VI. Again, the proposed system outperforms the commercial systems. The missing characters are marked by an asterisk in red color.

C. Mud Affected Plates

There are certain LPs badly affected due to aging or covered by mud, and plate detection becomes challenging in such cases. We show some examples in Table VII. Our system not only detects these plates but also recognizes the text. Only in the last plate, which is challenging even for the human eye, our system made one mistake out of seven characters. The other methods could not even detect that plate.

It would be interesting to see the performance of our system on English and Arabic parts of the LPs separately and combined. Fig. 15 shows the accuracy in all three scenarios for each of the 200 plates in the test dataset. Note that one plate has seven alphanumerics. If all of them are correctly recognized, we consider the accuracy to be 1. If 6 of them are recognized correctly, the accuracy is 6/7, and so on. It can be seen in the figure that the combined results, except for a few cases, could recognize all the characters correctly.

We also present the results of the individual alphanumeric recognition in Table VIII. The first column shows the method used for recognition. In 200 plates in our test dataset, there are 1400 characters in total. The second column shows how many of these were recognized correctly. The third column shows the number of cases when all seven alphanumerics in the plate were correctly recognized. This also gives the system accuracy in perfectly recognizing plates, which is shown in the last column as a percentage. The rest of the columns further break down the cases when less than seven alphanumerics were recognized correctly. It can be seen that the proposed system identified 194 plates out of 200 perfectly while making one mistake in five plates and two mistakes in one plate. The other methods performed much poorly in comparison. The closest competitor Sighthound could recognize only 110 plates.

TABLE VI. RESULTS OF DETERIORATED ALPHANUMERICS

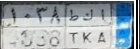
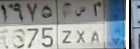
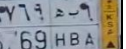
Model				
Sighthound	*7806HJ	*038TKA	*875ZXA	6*69HBA
OpenALPR	*806ZHJ	**38TKA	*375ZXA	6769HBA
Plate Recognizer	*806AHJ	*588TKA	*375ZXA	6*69HBA
Proposed	1806XHJ	1038TKA	1975ZXA	6769HBA

TABLE VII. RESULTS OF MUD AFFECTED PLATES

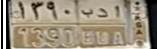


Model			
Sighthound	LP Not Detected	LP Not Detected	LP Not Detected
OpenALPR	LP Not Detected	LP Not Detected	LP Not Detected
Plate Recognizer	LP Not Detected	P9666VB	LP Not Detected
Proposed	1390BDA	2966GTB	79B8*UA

TABLE VIII. OVERALL RECOGNITION RESULTS USING CHALLENGING DATASET OF 200 LPs

Method	Correctly recognized characters (out of 200x7=1400)	Character recognition accuracy	Correctly recognized plates (out of 200)	Plate recognition accuracy
Proposed (English Only)	1106	79%	83	41.5%
Proposed (Arabic Only)	1303	93.1%	120	60%
Proposed (Combined)	1393	99.5%	194	97%
OpenALPR	910	65%	55	27.5%
Plate Recognizer	1030	73.6%	78	39.5%
Sighthound	1015	72.5	110	55%

V. CONCLUSION

This paper proposed a CNN model that can recognize English and Arabic text in the license plates used in Saudi Arabia. The two were combined using a proposed algorithm to correctly restore missing or wrongly read alphanumerics in either of the strings. These methods successfully recognized the license plates where commercially available solutions OpenALPR, Plate Recognizer, and Sighthound failed. The proposed system is computationally efficient and works in real-world unconstrained situations.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number MoE-IF-G-20-12.

REFERENCES

[1] V. R. Greati, V. C. T. Ribeiro, I. M. D. da Silva, and A. de Medeiros Martins, "A Brazilian license plate recognition method for applications in smart cities," in 2017 IEEE First Summer School on Smart Cities (S3C), 2017, pp. 43-48.

- [2] A. K. Haghghat, V. Ravichandra-Mouli, P. Chakraborty, Y. Esfandiari, S. Arabi, and A. Sharma, "Applications of deep learning in intelligent transportation systems," *Journal of Big Data Analytics in Transportation*, vol. 2, no. 2, pp. 115-145, 2020.
- [3] T. Björklund, A. Fiandrotti, M. Annarumma, G. Francini, and E. Magli, "Robust license plate recognition using neural networks trained on synthetic images," *Pattern Recognition*, vol. 93, pp. 134-146, 2019.
- [4] R. Laroca et al., "A robust real-time automatic license plate recognition based on the YOLO detector," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1-10.
- [5] M. Spanu et al., "Smart Cities Mobility Monitoring through Automatic License Plate Recognition and Vehicle Discrimination," in *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2021, pp. 1-6.
- [6] J. Li, H. Van Zuylen, Y. Deng, and Y. Zhou, "Urban travel time data cleaning and analysis for Automatic Number Plate Recognition," *Transportation Research Procedia*, vol. 47, pp. 712-719, 2020.
- [7] Z. Li et al., "License Plate Detection and Recognition Technology for Complex Real Scenarios," in *International Conference on Intelligent Computing*, 2020, pp. 241-256.
- [8] J. Shashirangana, H. Padmasiri, D. Meedeniya, and C. Perera, "Automated license plate recognition: a survey on methods and techniques," *IEEE Access*, vol. 9, pp. 11203-11225, 2020.
- [9] Y. Jamtsho, P. Riyamongkol, and R. Waranusat, "Real-time Bhutanese license plate localization using YOLO," *ICT Express*, vol. 6, no. 2, pp. 121-124, 2020.
- [10] S. Montazzolli and C. Jung, "Real-time brazilian license plate detection and recognition using deep convolutional neural networks," in *2017 30th SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, 2017, pp. 55-62.
- [11] S. Alghyaline, "Real-time Jordanian license plate recognition using deep learning," *Journal of King Saud University-Computer Information Sciences*, 2020.
- [12] V. Khare et al., "A novel character segmentation-reconstruction approach for license plate recognition," *Expert Systems with Applications*, vol. 131, pp. 219-239, 2019.
- [13] M. Lin, L. Liu, F. Wang, J. Li, and J. Pan, "License Plate Image Reconstruction Based on Generative Adversarial Networks," *Remote Sensing*, vol. 13, no. 15, p. 3018, 2021.
- [14] Z. Li, G. Han, S. Xiao, and X. Chen, "MAP-based single-frame super-resolution image reconstruction for license plate recognition," in *2009 International Conference on Computational Intelligence and Software Engineering*, 2009, pp. 1-5.
- [15] Y. Chen et al., "Image super-resolution reconstruction based on feature map attention mechanism," *Applied Intelligence*, pp. 1-14, 2021.
- [16] H. Li and C. Shen, "Reading car license plates using deep convolutional neural networks and LSTMs," *arXiv preprint arXiv:1606.05610*, 2016.
- [17] R. M. Khoshki and S. Ganesan, "Multiscale adaptive nick thresholding method for alpr system," *Entropy*, vol. 4, no. 10, 2015.
- [18] M. M. Mokji and S. A. Bakar, "Adaptive thresholding based on co-occurrence matrix edge information," in *First Asia International Conference on Modelling & Simulation (AMS'07)*, 2007, pp. 444-450.
- [19] SightHound. (September 2, 2020). Commercially available tool. Available: <https://www.sighthound.com/>.
- [20] PlateRecognizer. (November 21, 2020). Commercially Available Tool. Available: <https://platerecognizer.com/>.
- [21] OpenALPR. (December 22, 2020). Commercially Available Tool. Available: <https://github.com/openalpr/openalpr>.
- [22] R. Chen and Y. Luo, "An improved license plate location method based on edge detection," *Physics Procedia*, vol. 24, pp. 1350-1356, 2012.
- [23] J.-Y. Sung and S.-B. Yu, "Real-time Automatic License Plate Recognition System using YOLOv4," in *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, 2020, pp. 1-3.
- [24] I. R. Khan, S. T. A. Ali, A. Siddiq, M. M. Khan, M. U. Ilyas, S. Alshomrani, S. Rahardja, "Automatic License Plate Recognition in Real-World Traffic Videos Captured in Unconstrained Environment by a Mobile Camera," *Electronics*, vol. 11, no. 9, pp. 1408, 2022.
- [25] I. Kilic and G. Aydin, "Turkish vehicle license plate recognition using deep learning," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 2018, pp. 1-5.
- [26] S. E. Ozbay Ergun, "Automatic vehicle identification by plate recognition," *World Academy of Science, Engineering Technology*, vol. 9, no. 41, pp. 222-225, 2005.
- [27] Steve. (October 20, 2019). Types of License Plates in Saudi Arabia. Available: <https://lifeinsaudiArabia.net/types-of-number-plates-in-saudi-arabia/>.
- [28] G. Jocher. (July, 2019). YOLOV5, An Open-Source Detection Model Repository Available: <https://github.com/ultralytics/yolov5>.
- [29] T.Lin. (March 29, 2018). LabelImg, Annotation Tool. Available: <https://github.com/tzutalin/labelimg>.
- [30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.
- [31] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10781-10790.
- [32] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390-391.
- [33] ProgrammerSought. (July, 2020). YOLOV5 learning summary.
- [34] J. Wen-ping and J. Zhen-cun, "Research on early fire detection of Yolo V5 based on multiple transfer learning," *Fire Science Technology*, vol. 40, no. 1, p. 109, 2021.
- [35] A. Khan, A. Sohail, U. Zahoor, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455-5516, 2020.