A Robust License Plate Detection and Recognition Framework for Arabic Plates with Severe Tilt Angles

Khaled Hefnawy, Ahmed Lila, Elsayed Hemayed, Mohamed Elshenawy
Communication Department
Zewail City, Egypt

***Abstract*—This paper addresses the challenge of accurately detecting and recognizing Arabic license plates, particularly those subjected to severe tilt angles. It presents a robust license plate detection and recognition framework that consists three main steps: plate detection and segmentation, plate perspective correction, and vehicle number recognition. In the first step, a mask R-CNN model is used to detect the plate location, providing pixel-wise labels of identified plates' areas. Following this, a perspective correction technique is used to obtain a clear and rectangular image of each license plate in the image. Lastly, the framework employs a Bidirectional Long Short-Term Memory (Bi-LSTM) model for accurate vehicle number recognition. The framework's efficacy is demonstrated through its application to build a plate recognition system tailored for Egyptian license plates. The system was tested on a dataset collected from campus gate cameras at Zewail city of science and technology, achieving a character accuracy of 97%.**

Keywords*—*License plate detection; license plate recognition; feature extraction; Mask R-CNN; object detection

I. Introduction

The Automatic License Plate Recognition (ALPR) systems have become integral to modern transportation and smart city initiatives [1]. The rapid evolution of surveillance systems has increased the reliance on and demand for ALPR to analyze and process the extensive amounts of camera feeds generated daily. An efficient and reliable ALPR system is critical in many smart city applications including toll collection, safety monitoring, managing border crossings and parking management.

A typical ALPR system consists of three main stages: plate detection, character segmentation, and character recognition [2]. The process begins with plate detection, which locates the plate within the image. This is followed by character segmentation, which isolates each character or digit within the plate. Identified characters are then processed via a character recognition model that reads the plate number.

The incorporation of character segmentation and recognition in ALPR systems faces significant challenges when adapted to new environments [3]. Key challenges include different camera orientations, changes in illumination, and variations in image quality. For instance, models trained on license plates positioned directly in front struggle with footage from cameras mounted at high elevations or at acute angles to moving vehicles, significantly diminishing their recognition accuracy [4].

Furthermore, license plate designs vary across countries, with notable variations in size, language, color, and font. Accordingly, a one-size-fits-all system is impractical and there is a necessity for adaptable frameworks to guide the implementation and customization of these systems across diverse contexts [5].

Several recent research efforts have been proposed to address the above issues. Lin and Li [6] suggest the use of three-stage ALPR system that utilizes YOLOv2 for vehicle localization, in the first stage, and license plate localization, in the second stage. The final stage employs a mask R-CNN to detect each character in the plate. A similar approach is used in [5], where the authors suggest a three-stage system. The first stage employs Faster R-CNN to localize vehicles, followed by the use of morphological operations to localize license plates within the detected vehicles. The final stage utilizes a deep learning model for character recognition. Another approach [7], focused on license plate detection, uses a faster R-CNN for vehicle detection, morphological filtering for license plate detection. These efforts focus on recognition of non-Arabic plates.

Fewer research efforts have focused on Arabic plate recognition [8]. The nature of the Arabic characters, including its right-to-left writing system, complicate segmentation and recognition tasks. Additionally, model performance may be affected by environmental factors such as lighting conditions, plate obfuscation, and the presence of dust or damage on the plates. While using advanced deep learning techniques have shown promise in overcoming these challenges [9], most of the proposed techniques require extensive, diverse training datasets and careful tuning to build robust models. Further research is essential to enhance the robustness of Arabic license plate recognition systems.

This paper presents a robust framework for Arabic license plate detection and recognition. The framework is structured into three key steps. In the first step, the framework employs a Mask R-CNN model [10] for precise localization and segmentation of license plates. This is followed by a technique for perspective correction, ensuring each plate is presented in a clear and standardized rectangular format. Finally, the system incorporates a Bidirectional Long Short-Term Memory (Bi-LSTM) model, specifically employed to achieve accurate recognition of Arabic plate letters and numbers.

The system was trained using two datasets: the first dataset is a synthetic dataset, available on Kaggle [11], which consists of 5k of Arabic license plate images. The second dataset, which includes 2,712 real images from 300 unique cars, was gathered at a toll gate in Egypt. The dataset was originally created by Elnashar et al. [12]. We extended it by adding annotations of the license plate letters and numbers. The efficacy of the

recognition was validated using a test dataset of real 140 images collected from cameras at the entrance of Zewail City of Science and Technology. The system achieved a character accuracy of 97%.

II. Related Work

As discussed earlier, the proposed framework has three main stages: license plate detection, perspective correction, and license plate recognition. This section reviews some prominent research efforts in license plate detection and recognition. In our review, we focus on Arabic license plate detection and recognition, as this is the scope of this work.

A. License Plate Detection

Traditional license plate detection methods have utilized image processing techniques, such as the Sobel edge detector [13] and the Hough transformation [14], to detect lines and edges in the given image, providing a way of identifying the license plate's boundaries. These methods, however, are prone to errors from noisy edges and complex images.

Plate texture is also used as a good detection indicator of the plate. Techniques such as Gabor filter [15] and Wavelet transform [16] have been used to detect the plate using its texture. Others used a decision tree with adaptive boosting on Haar-like characteristics [17]. This algorithm performed well in multiple camera orientations and lighting conditions. It achieved an accuracy of 94.5%.

Recently, deep learning techniques have been used extensively in license plate detection mechanisms. For instance, YOLOv3 was used to detect plates on multi-style Egyptian license plates [12]. This approach achieved 99.2% plated detection accuracy on the new standardized Egyptian license plates and 96.8% for the older versions of Egyptian plates. Another example is the detection algorithm in [18]. The algorithm uses a region-based convolutional neural network (R-CNN) to achieve dual objectives: plate detection and estimation of the four corners of the plates. Subsequently, the identified points were utilized to perform an affine transformation on the plates, correcting any tilt angles present.

B. License Plate Recognition

Automatic Arabic License Plate Recognition (AALPR) systems can be classified into two main approaches: segmentation-based approach [19] and segmentation-free approach [19] & [20]. The segmentation-based approach segments individual license plate characters and then detects each character using an OCR model. Elsaid, et al. [21] proposed a segmentation-based approach trained on Arabic and Indian numerals and Arabic alphabets on Saudi Arabian license plates. The proposed pipeline has five stages: license plate preprocessing, license plate localization, character segmentation, features extraction, and character recognition. It was tested on 470 license plates with different effects such as skewness and noise, achieving 96% segmentation accuracy and 94.7% recognition rate.

Antar et al. [22] proposed a method for detecting and recognizing vehicle license plates. A canny edge detector is used for the detection module, combined with other noise reduction techniques. The recognition module uses a masking technique to locate regions of interest in the license plate and applies a character recognition model to classify each region. The model achieved 96% accuracy for English and 92.4% accuracy for Arabic. Sarfraz et al. [13] performed character segmentation and normalization, then template matching with Arabic characters.

Shehata et al. [19] proposed four systems to recognize license plates. The first system is a segmentation-based approach and has three main modules: license plate extraction, character segmentation module, and character recognition module. The character recognition component employs a KNN classifier to assess the resemblance between the segmented characters and the reference characters stored as ground truth. The second system has the same modules as the first system but uses Deep CNN for feature extraction and Deep CNN for character recognition. The system was tested on 300 license plates and got 0.95 recall, 0.95 precision, and a character recognition accuracy of 0.99 which lead to license plate accuracy rate of 0.95.

The third system in Shehata et al.'s work is a segmentation-free approach. The model is based on two classical machine learning modules: license plate feature extraction and license plate recognition. The system was tested on 100 license plates and got 0.76 recall, 0.85 precision, and accuracy rate of 0.90. The fourth system is another free-segmentation approach with two deep learning modules: plate detection and plate recognition. The system was tested on 1000 license plates and got 0.89 recall, 0.91 precision, and accuracy rate of 0.93.

III. The License Plate Detection and Recognition Framework

As discussed earlier, the framework has three main stages: the plate detection and segmentation step, plate perspective correction, and plate recognition. A system implementing the proposed framework is shown in Fig. 1. The proposed system has four main components: the segmentation model, the corner estimation, the plate warping model, and the plate recognition model.

The first stage of the framework, plate detection and segmentation, is implemented via the license plate segmentation model shown in Fig. 1. The output of this model is pixel-wise segmentation of the detected plate, as indicated in the Figure. It should be noted that the model is trained to detect the plate directly without detecting the vehicle first.

The second stage, plate perspective correction, is implemented via the corner estimation and plate warping components. The segmented plate is passed through a series of operations to estimate the corner of the plate's rectangular area, namely edge detection, contour fitting, and contour fine-tuning. A homography transformation is then calculated and applied in the plate warping component to correct the plate perspective.

The corrected plate is passed to the recognition module in the last stage to get the recognized characters. The details

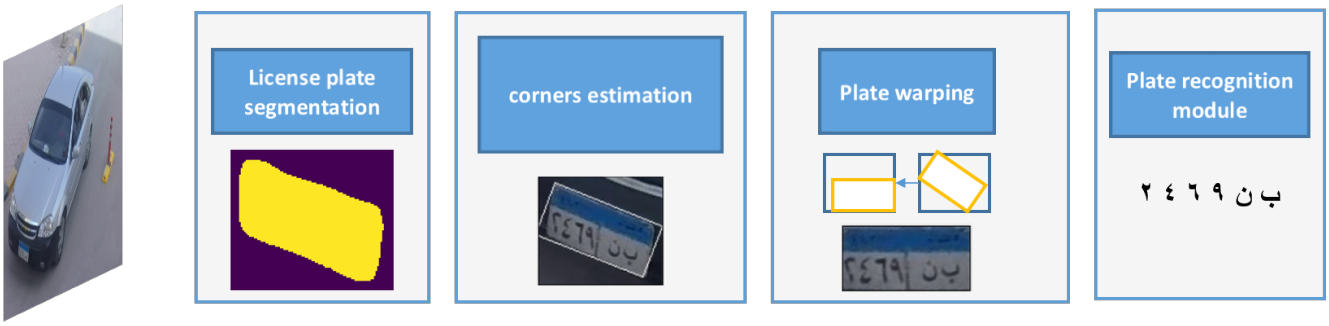

Fig. 1. System overview.

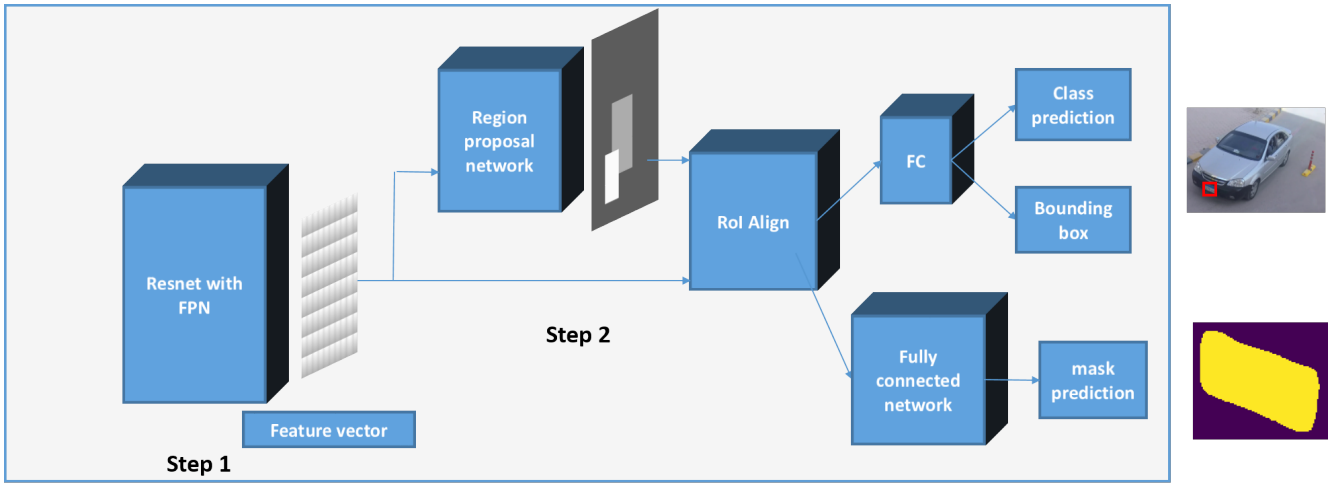

Fig. 2. Mask R-CNN.

about each model are discussed in the following subsections.

A. Plate Detection and Segmentation

We used Mask R-CNN [10] to detect the plate and its bounding box. Mask R-CNN is an extension for faster R-CNN [23] where a fully connected head is added to the faster R-CNN network to get additional output for the bounding boxes and classes. It gives a pixel-wise annotation for the image, which offers highly accurate labels. The segmentation enables us to get arbitrary shapes that describe precisely objects of interest. Mask R-CNN is shown in detail in Fig. 2.

Faster R-CNN is composed of two main parts. The first part proposes potential object bounding boxes using a region proposal network (RPN). In the second part, a region-of-interest (RoI) Pool layer extracts features for each potential bounding box, classifies it, and regresses its bounding box.

Feature extraction is done using ResNet [24] and Feature pyramid network (FPN) [25]. Features from ResNet are passed to RPN, which uses a sliding window to perform convolutions on different positions of the image using multiple anchors to detect the license plates. FPN passes different scales of the input image to the RPN. Using different scales of the input image makes the network more robust at detecting small license plates.

Mask R-CNN has the same structure as faster R-CNN, with the first part being the same (RPN). However, in the second part, a binary mask is formed parallel to class and bounding box predictions. This parallel computing simplified the training process greatly by combining multi-task loss and optimizing them concurrently on each potential object of interest. Multi-task loss is formed as shown in Eq. (1)

$$L = L_{cls} + L_{box} + L_{mask} \tag{1}$$

where L_{cls} & L_{box} are the classification and bounding-box loss, respectively, as defined in faster R-CNN. The mask network has dimensions of m^2 for each potential object of interest. For each pixel, a sigmoid activation function is computed. L_{mask} is computed by taking the mean of the cross entropy loss over all the pixels.

B. Corner Estimation

After the plate segmentation extraction step, we process the image to estimate the corners of the segmented plate image to be used in correcting the plate perspective as shown in Fig. 3. First, we apply edge detection to help get the minimum rotated bounding box. Then, we use a canny algorithm [26], which passes the images through multiple steps to find the edges.

To get the rotated bounding box coordinates, we use a rotating calipers algorithm [27] to find the minimum bounding box that fits the segmentation polygon output. The rotated

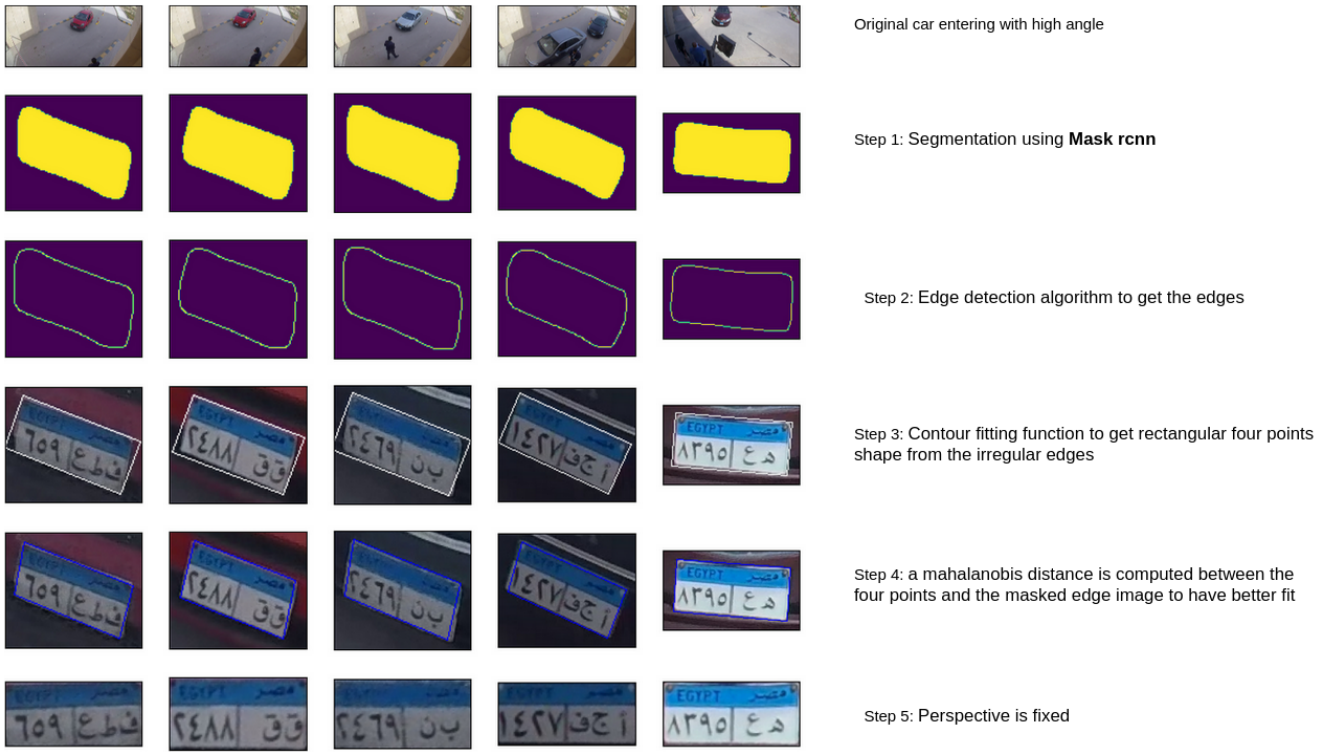

Fig. 3. Perspective correction steps.

caliper algorithm starts with fitting a rectangle having one side aligned with the target polygon and a set of the polygon vertices. It updates the coincident edge and the vertices until it reaches the minimum fitting rotated bounding box efficiently in just O(n), where n is the number of polygon vertices.

C. Plate Warping

Due to the tilted angle of the camera, the detected plates do not have a rectangular shape, which affects the character recognition task. In this step, we fix the plate's perspective projection, improving the plate character recognition accuracy. The perspective transformation changes the viewpoint of the image, e.g., the angle, lengths, and geometry of the lines, but it keeps the collinearity of the points. Hence, we can correct the perspective of the plate by computing the transformation matrix needed to translate points from their actual view to the front-facing view. The transformation uses the actual viewpoints and the targeted points to get the matrix. This operation can be formulated using the transformation matrix in Eq. (2)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2}$$

where constants a,b,c,d control rotation and scaling transformation, e,f control the translation vector, and g,h are the projection vectors.

We use the four input points (estimated corners) and their targeted location points (rectangular shape) to have eight equations that could be solved simultaneously to get the transformation constants.

D. Plate Character Recognition

Because of its excellent performance in different detection and recognition vision tasks, we initially trained and tested the YOLOv5 model for character recognition. However, the results could have been better after fine-tuning the model, as discussed in the experimental results section. Thus, we developed a new model for character recognition. The plate character recognition proposed module has two main steps: license plate feature extraction and character predictions. We will show the components of each module and how they are combined and trained at once to best solve the problem of limited datasets. We use a stack of CNN and RNN layers so that it can perform character-level inferences without the need for an explicit step for character segmentation.

License plate features extraction: Inspired by the VGG network [28], a CNN is used for character feature extraction. The ReLU activation function is used after each convolutional layer to add nonlinearity to the network. We used batch normalization for regularization. He initialization [29] was used to prevent the vanishing gradient problem and speed up the training process. The channel information was gradually reduced after each CNN layer using the maxPool layer to the required timesteps. Afterward, the feature extraction output is passed to the RNN layers using the Bi-directional LSTM.

Characters prediction: The developed model uses RNN to learn the sequential relations found in the license plates

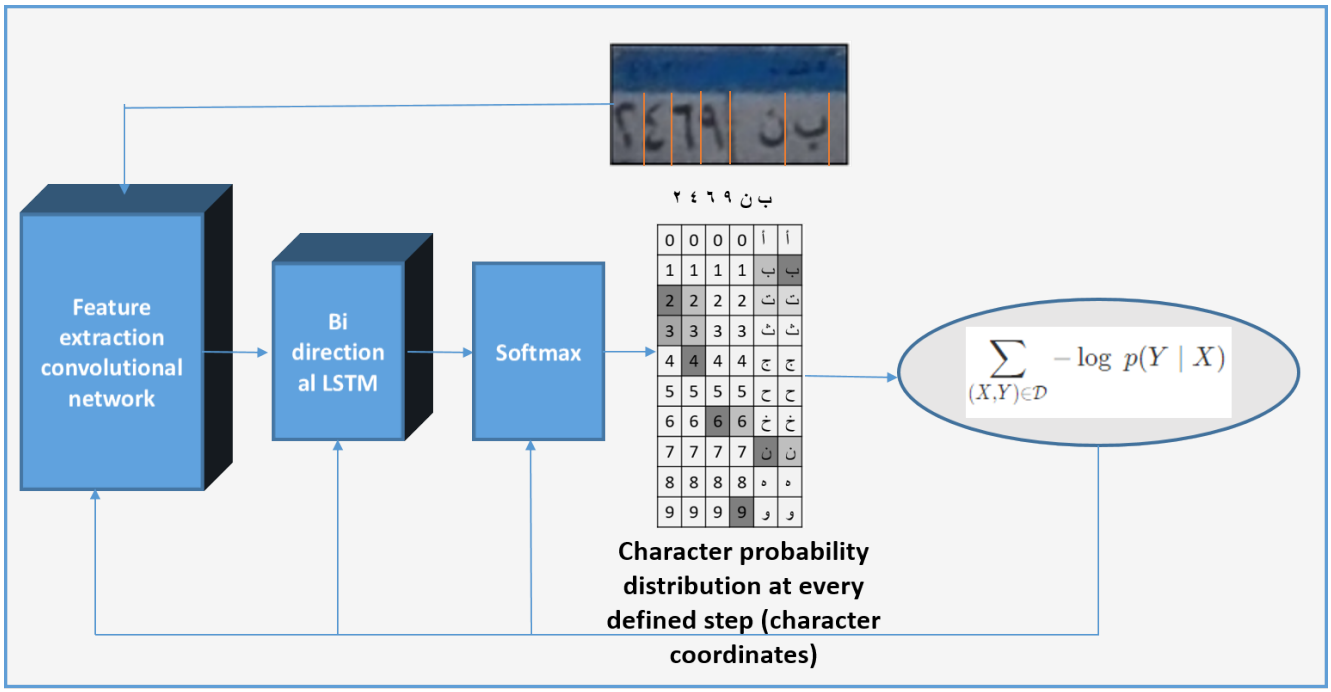

Fig. 4. Plate recognition module.

combined with the CTC loss presented in Eq. (3).

$$\mathcal{L}_{CTC} = -\log p(\mathbf{y}|\mathbf{x}) = -\sum_{\mathbf{s}\in\mathcal{B}(\mathbf{y})} \log p(\mathbf{s}|\mathbf{x}) \tag{3}$$

where

- $\mathbf{y}$ is the target sequence, which is the ground truth license plate characters.
- $\mathcal{B}(\mathbf{y})$ is the set of all possible label sequences that can be derived. It includes all possible combinations of the target sequence with additional blank labels inserted between and after the characters, representing all possible alignments of the characters in the input sequence.
- $p(\mathbf{s}|\mathbf{x})$ is the probability of label sequence $\mathbf{s}$ given the input sequence $\mathbf{x}$, as computed by the CTC model. This probability is estimated by the neural network during training.
  The loss is computed as the negative log-likelihood of the target sequence, given the input sequence and the probabilities of all possible label sequences.

The whole network of the CNN and RNN is trained end to end using CTC loss. This takes care of extracting the implicit rules that are present in the license plates. For instance, the Egyptian license plate uses a structure in which, if the plate is read from left to right, the plate has four digits followed by three characters. By learning the sequence, the network can predict getting a digit in the first four values and characters in the last three.

Those learned rules direct the training of the feature extraction in the CNN layer. The whole network parameters are trained and optimized at once from the CTC loss. The loss is between labels, and targets are backpropagated to update all the parameters. Finally, a dense layer maps the output of the RNN network to the predicted category. We use the SoftMax function to obtain the probability of the correct character as shown in Fig. 4

The CTC loss function is used to decode sequential information to align plate characters in a segmentation-free way. This approach also has the advantage of not needing bounding box annotation for every character, simplifying the learning process. An annotator only needs to record the character of every license plate on a sheet.

CTC technique allows the training of sequence-to-sequence models, such as recurrent neural networks (RNNs), with variable-length inputs and outputs. It works by defining a set of possible alignments between the input and output sequences, called the "blank label" set, and computing the likelihood of each alignment given the input sequence.

In terms of time steps, CTC works as follows:

1) At each timestep, the RNN generates a probability distribution over all possible labels, including a special "blank" label.
2) For each possible label sequence, the CTC algorithm computes the probability of that sequence given the input sequence.
3) The CTC algorithm then sums the probabilities of all possible label sequences that can be derived from the target sequence by inserting, deleting, or repeating the blank label.
4) The CTC loss function is defined as the negative log-likelihood of the target sequence given the input sequence and the probabilities of all possible label sequences.

5) During training, the RNN's parameters are updated to minimize the CTC loss.
6) In the decoding stage, the probabilities generated by the RNN are used to find the most likely label sequence. In summary, CTC works by defining a set of possible alignments between the input and output sequences, computing the likelihood of each alignment given the input sequence and minimizing the negative log-likelihood of the target sequence.

The training dataset is fed to the model with a batch size of 256 images. We used the Adam optimizer for faster convergence and better performance using gradient descent and momentum optimization.

IV. Experimental Result

This section discusses the experimental results, starting with a description of the datasets used in the training and testing of the developed models.

A. Datasets

Accurate datasets are necessary for training ALPR deep learning models. There are different aspects to be considered in the ALPR datasets, including environment-related aspects such as illuminations and background; camera point-of-view-related aspects such as the pose of the camera (frontal or upper view), rotation and skew Coefficients; and plate-related aspects such as color distribution, location of the plate with respect to the car, language and font styles.

Multiple datasets have been used in this research to improve the system's overall accuracy. These datasets are 1) the Macathon competition dataset on kaggle [11], 2) the toll dataset [12] and 3) a proprietary dataset collected from CCTV cameras on campus. Table I summarizes the distributions of each dataset.

TABLE I. Datasets Summary

	Datasets		
	Synthetic	Toll	University
Year	2022	2022	2022
number of images	7k	2k	140
LP size	376 x 172	68 x 32	100 x 50
LP angle	Frontal	frontal	Oblique

1) Kaggle macathon competition dataset: The Machathon competition dataset comprises synthetically modified 5k images of Arabic license plates. This dataset has annotations for the plate's characters, including characters' bounding boxes. The images are generated with a high resolution that isn't realistic in real-life scenarios. Samples of the dataset are shown in Fig. 5, and the distribution of the characters is shown in Fig. 6. The distribution indicates that the classes are unbalanced and digits are more common than letters.

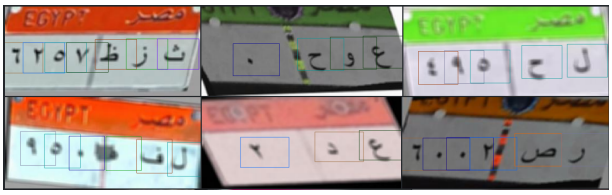

Fig. 5. Macathon competition dataset samples.

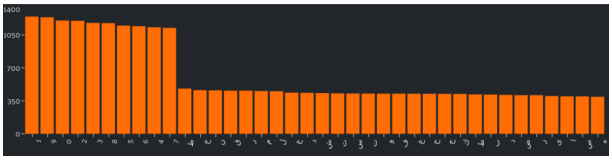

Fig. 6. Macathon competition dataset character distribution.

After inspection of the character bounding boxes, it was noticed that some bounding boxes may cover more than one character, which causes problems if used without correction. As such, we used a bounding box correction process, summarized visually in Fig. 7, to improve the dataset's quality. First, each bounding box is extracted from the image. A fixed constant, decided experimentally, is removed from each margin so the bounding box encloses only one character. Secondly, a K-means segmentation is used to extract three clusters in each bounding box. The bounding box around the largest cluster is considered the corrected bounding box. Finally, annotations are fixed manually to account for errors that could occur due to noise. Samples of the corrected annotations are shown in Fig. 8.

2) Toll dataset: The toll dataset [12], includes real images collected at a toll gate in Egypt. Fig. 9 displays samples from the toll dataset. An imbalance is observed in the toll dataset, as demonstrated in the character distribution in Fig. 10.

The car images are taken from a frontal view camera facing the entering cars, which means the dataset does not present the tilted angle challenge addressed in this paper. The dataset contains 2712 images, corresponding to 300 unique cars, with the license plate at varying proximities from the camera and in different qualities. For the purpose of character recognition, all images were manually labeled with the correct characters and digits shown on the license plate.

3) The Arabic License Plate Recognition (ALPR) Dataset: The Arabic License Plate Recognition (ALPR) dataset consists of two hours of videos captured from two gate cameras at the entry and exit gates of campus. This dataset consists of 140 images that the model has not seen. Samples of this dataset are shown in Fig. 11. As seen in the figure, the dataset is challenging. Due to the pose and height of the camera, all plates are tilted. Some plates have different illuminations, and others are distorted.

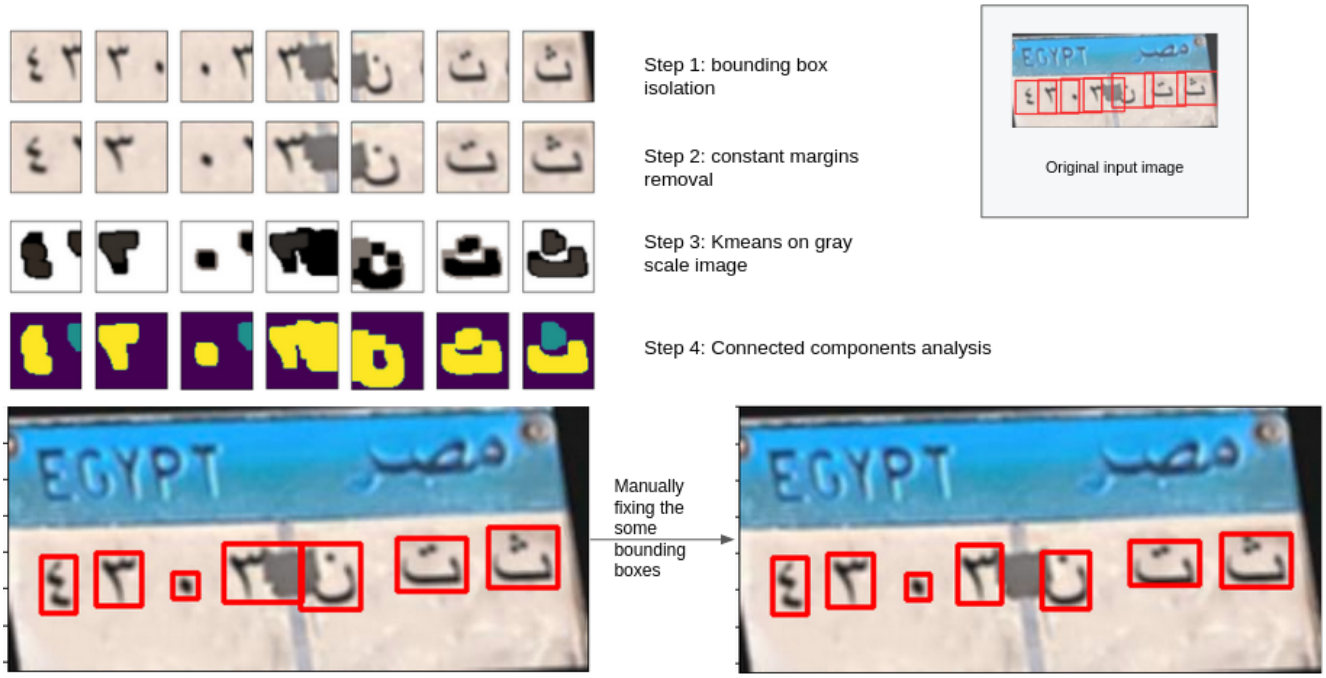

Fig. 7. Synthetic dataset correction steps.

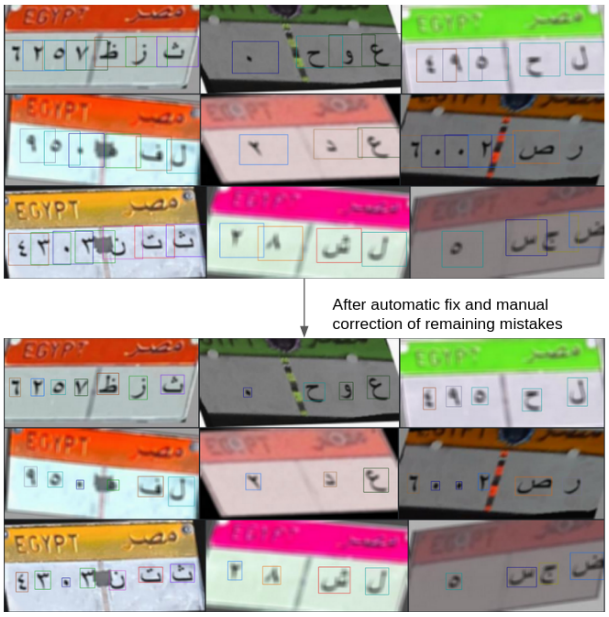

Fig. 8. Synthetic dataset before and after corrections.

Fig. 9. Sample of the toll dataset.

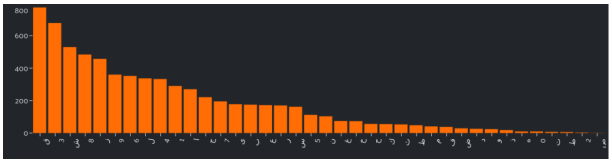

Fig. 10. Toll dataset character distribution.

To enrich the dataset for the character recognition task, an augmentation process was used to increase the dataset 10 times. Plate images were resized to dimensions of 32 pixels in height and 128 pixels in width. Experiments were conducted with different kinds of augmentations, such as random brightness, contrast, image quality, and saturation. Additionally, the recommended YOLOv5 augmentations were used, which include HSV color space augmentation, Letterbox, Mixup, and Random perspective.

B. Results

The plate recognition mechanism was trained on the Kaggle Machathon competition dataset. The dataset was split into three sets for training, validation, and testing. The training dataset was used to extract the patterns of the letters. The validation dataset was used to choose the right parameters of the neural network. Finally, the testing dataset was used to provide unbiased accuracy results. The recognition is tested using two models YOLOv5 and the VGG network supported

Fig. 11. Samples of the ALPR testing dataset and its challenges.

by bidirectional LSTM layers. Table II shows the results when these models were applied to the validation dataset.

Three metrics are used to evaluate the plate recognition model:

1) character accuracy: This metric represents how many characters were recalled correctly
2) Identical 0: This metric represents how many plates were identified completely correctly
3) Identical 1: This metric represents how many plates were identified with a one-character error

TABLE II. CHARACTER RECOGNITION ACCURACY FOR THE VALIDATION DATASET

Model	Character Accuracy	I0	I0+I1
Yolov5	96.89%	86.43%	98.12%
VGG11 - BI LSTM	99.54%	97.9%	100%

Then, both models were tested on the Campus dataset. Table III shows the accuracy of both models.

TABLE III. CHARACTER RECOGNITION ACCURACY FOR THE TESTING DATASET FROM THE CAMPUS

Model	Character Accuracy	I0	I0+I1
VGG11 - BI LSTM using CTC loss	82.29%	27.9%	67.9%
VGG11 - BI LSTM more training data	94%	73.5%	92.1%
VGG11 - BI LSTM fine-tuned on annotated toll dataset	97%	86%	96.4%

As seen from the table, VGG using bidirectional LSTM surpasses YOLOv5. The list below discusses the performance of the two models.

1) VGG was more accurate than YOLOv5 in mapping the relationship between letters.
   a) There is some correlation between two letters that should appear together in a certain province or area
   b) There is some correlation between letters and digits, meaning that if two or a maximum of three letters appear, the coming characters are sure to be digits. VGG can map this very well, while YOLOv5 makes mistakes in this. For example, the number 4 in Arabic may be misrecognized by letter ع .
2) YOLOv5 requires a lot of data to generalize well on unseen data
3) These results are satisfactory considering the limited availability of training data. The accuracy is expected to increase as the model works and collects more data. For the time being, accuracy can be improved by using a voting mechanism for the recognition results of several images of the same plate. Three stages were shown in training VGG network; the last stage with most data showed the best results on the test set.

Fig. 12 shows a sample of the mistakes encountered when using YOLOv5 but were correctly handled by the VGG. In this case, the number 4 is misrecognized by غ or ع. Fig. 13 shows two plates that were recognized entirely by VGG, but only two letters were recognized when using YOLOv5.

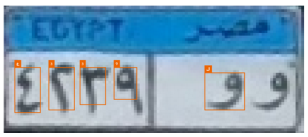

Fig. 12. Some samples not detected by YOLOv5 but were correctly detected by VGG.

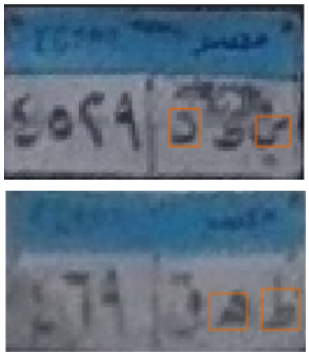

Fig. 13. Challenging samples that were correctly recognized by VGG but not YOLOv5.
Bounding boxes represents the detection of YOLOv5.

V. Conclusion

This study has demonstrated that algorithms with inherent automatic rules extraction, combined with effective preprocessing, outperform deep learning methods for license plate recognition when working with limitations in both the number of instances and the size of images. The results indicate the importance of preprocessing and the potential of automatic context-understanding approaches in applications where data are limited or expensive to obtain. By continuing to refine the methodology and expand the dataset, the aim is to further improve the accuracy and efficiency of license plate recognition systems.

VI. Future Work

In future work, the plan is to further enhance the license plate recognition system by increasing the dataset size, which is expected to improve the model's performance. Additionally, integrating an annotation tool into the system will facilitate the creation and management of annotated data, streamlining the training process for the models. These improvements will help better understand the impact of larger datasets and more sophisticated preprocessing techniques on license plate recognition accuracy. It will also provide the ability to train more deep models that can ingest large datasets.

VII. Acknowledgment

We would like to express our appreciation for the funding provided by the Science, Technology and Innovation Funding Authority (STDF) in Zewail City, which was crucial for the success of this project.

References

[1] R. A. Lotufo, A. D. Morgan, and A. S. Johnson, "Automatic number-plate recognition," in IEEE Colloquium on Image Analysis for Transport Applications, Feb 1990, pp. 1–6

[2] J. Shashirangana, H. Padmasiri, D. Meedeniya, C. Perera, "Automated license plate recognition: a survey on methods and techniques," in IEEE Access, vol. 9, pp. 11203-11225, Dec. 2020.

[3] Kamal, Nada & Khudair, Enas. (2021). License Plate Tilt Correction: A Review. Engineering and Technology Journal. 39. 101-116. 10.30684/etj.v39i1B.1839.

[4] T.G. Kim, B.J. Yun, T.H. Kim, J.Y. Lee, K.H. Park, Y. Jeong, and H.D. Kim, "Recognition of vehicle license plates based on image processing." in Applied Sciences, vol. 11, , no. 14:6292, 2021, doi: 10.3390/app11146292.

[5] F. Sultan, K. Khan, Y.A. Shah, M. Shahzad, U. Khan, and Z. Mahmood, "Towards Automatic License Plate Recognition in Challenging Conditions." in Applied Sciences, vol. 13, no. 6:3956, 2023, doi: 10.3390/app13063956.

[6] H. Lin and Y. Li, "A License Plate Recognition System for Severe Tilt Angles Using Mask R-CNN," 2019 International Conference on Advanced Mechatronic Systems (ICAMechS), Kusatsu, Japan, 2019, pp. 229-234, doi: 10.1109/ICAMechS.2019.8861691.

[7] Z. Mahmood, K. Khan, U.Khan, S.H. Adil , S.S.A. Ali , M. Shahzad, "Towards Automatic License Plate Detection," in Sensors, vol. 22, no. 3:1245, 2022, doi: 10.3390/s22031245

[8] G. Alkawsi, Y. Baashar, A. A. Alkahtani, T. S. Kiong, D. Habeeb and A. Aliubari, "Arabic Vehicle Licence Plate Recognition Using Deep Learning Methods: Review," 2021 11th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 2021, pp. 75-79, doi: 10.1109/ICCSCE52189.2021.9530940.

[9] A. R. Youssef, A. A. Ali, and F. R. Sayed, "Real-time Egyptian License Plate Detection and Recognition using YOLO," in International Journal of Advanced Computer Science and Applications (IJACSA), vol. 13, no. 7, 2022.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961-2969.

[11] STP 2022, "Machathon 3.0," Kaggle, 2022. [Online]. Available: https://kaggle.com/competitions/machathon-3

[12] M. Elnashar, E. Hemayed and M. Fayek, "Automatic Multi-Style Egyptian License Plate Detection and Classification Using Deep Learning", 2020 16th International Computer Engineering Conference (ICENCO), 2020. Available: 10.1109/icenco49778.2020.9357371

[13] M. Sarfraz, M. J. Ahmed and S. A. Ghazi, "Saudi Arabian license plate recognition system," 2003 International Conference on Geometric Modeling and Graphics, 2003. Proceedings, 2003, pp. 36-41, doi: 10.1109/GMAG.2003.1219663.

[14] G. Heo, M. Kim, I. Jung, D.-R. Lee, and I.-S. Oh, "Extraction of car license plate regions using line grouping and edge density methods," in Proc. Int. Symp. Inf. Technol. Converg. (ISITC), Nov. 2007, pp. 37–42.

[15] H. Caner, H. S. Gecim, and A. Z. Alkar, "Efficient embedded neuralnetwork-based license plate recognition system," IEEE Trans. Veh. Technol., vol. 57, no. 5, pp. 2675–2683, Sep. 2008

[16] Y.-R. Wang, W.-H. Lin, and S.-J. Horng, "A sliding window technique for efficient license plate localization based on discrete wavelet transform," Expert Syst. Appl., vol. 38, no. 4, pp. 3142–3146, Apr. 2011.

[17] Q. Wu, H. Zhang, W. Jia, X. He, J. Yang, and T. Hintz, "Car plate detection using cascaded tree-style learner based on hybrid object features," in Proc. IEEE Int. Conf. Video Signal Based Surveill., Nov. 2006, p. 15.

[18] M. Dong, D. He, C. Luo, D. Liu, and W. Zeng, "A CNN-based approach for automatic license plate recognition in the wild," Procedings of the British Machine Vision Conference 2017, 2017.

[19] M. Shehata, M. T. Abou-Kreisha, and H. Elnashar, "Deep Machine Learning based Egyptian Vehicle License Plate Recognition Systems," In SpringerLink, 01-Jan-1970. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-36368-0_12.

[20] M. Fasha, B. Hammo, N. Obeid, and J. Widian, "A hybrid deep learning model for Arabic text recognition, " In arXiv.org, 04-Sep-2020. [Online]. Available: https://arxiv.org/abs/2009.01987.

[21] S. A. Elsaid, H. Alharthi, R. Alrubaia, S. Abutile, R. Aljres, A. Alanazi, and A. Albrikan, "Arabic real-time license plate recognition system," In arXiv.org, 24-Jul-2021. [Online]. Available: https://arxiv.org/abs/2107.11640.

[22] R. Antar, S. Alghamdi, J. Alotaibi, and M. Alghamdi, "Automatic number plate recognition of Saudi license car plates," in Engineering, Technology and Applied Science Research, vol. 12, no. 2, pp. 8266–8272, Apr-2022. [Online]. Available: https://www.etasr.com/index.php/ETASR/article/view/4727/2690.

[23] R. Girshick, "Fast r-cnn," Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440-1448.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770-77

[25] T.Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2117-2125.

[26] J. Canny, "A Computational Approach to Edge Detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[27] Toussaint, Godfried. (2000). Solving Geometric Problems with the Rotating Calipers. In Proceedings of IEEE MELECON'83. 83.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv.org, 10-Apr-2015. [Online]. Available: https://arxiv.org/abs/1409.1556. [Accessed: 16-Mar-2023].

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet Classification," arXiv.org, 06-Feb-2015. [Online]. Available: https://arxiv.org/abs/1502.01852. [Accessed: 16-Mar-2023].