STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–12.
Published online in International Academic Press (www.IAPress.org)

A Statistical Modeling and Evaluation Framework for YOLO-Based Models in Cross-Language License Plate Recognition: Arabic and Latin Alphabets

Israa Lewaaelhamd[1,*], Ahmed Elaraby[2]

[1]*Department of Business Administration, Faculty of Business Administration, The British University in Egypt, Cairo, Egypt.*
[2]*Department of Computer Science, Faculty of Computers and Information, QENA University, Qena 83523, Egypt.*

Abstract This study presents a statistical performance modeling framework for cross-language license plate recognition using YOLO-based models. License plate recognition is crucial for enforcing traffic regulations and improving road safety. A dataset of 1,834 images was analyzed to model and quantify the performance of two YOLO-based models, YOLOv5 and YOLOv8, using statistical metrics such as accuracy, precision, and recall. Recognizing license plates in diverse scripts, including Arabic and Latin alphabets, presents significant challenges. The modeling results indicate that YOLOv8 demonstrates superior performance, achieving an accuracy of 96.1% compared to 94.1% for YOLOv5. This study illustrates that a rigorous statistical modeling approach provides a robust framework for understanding, predicting, and comparing model performance, highlighting the potential of integrating deep learning with statistical methods for applied computer vision tasks.

Keywords Statistical modeling, Applications, Deep learning, Vehicle identification, YOLO-based models, Arabic license plate recognition, Statistical models, Neural networks.

AMS 2010 subject classifications 62H30, 62F99, 68T07, 68T45.

DOI: 10.19139/soic-2310-5070-3407

1. Introduction

In the last decade, machine learning algorithms, specifically supervised learning, have shown promising results in classification or detection problems in various tasks of computer vision. Using machine learning or, more recently, deep learning, we don't need to specify directly which features are important to differentiate between the object and its local background. Instead, we let the algorithm decide which part of the image is the best feature to represent the object. Supervised learning specifically needs annotated data as a training set to learn a mapping function from input X to output Y, or informally, to learn a decision function to map an input to its correct output. This trend also answered a research gap in the old methods where it was difficult to design a license plate recognition system for any different plate format and character pattern due to the language dependency of the handcrafted features. The automatic feature selection and machine learning method will adapt directly to the different characteristics of each license plate and its characters [1]. The early history of recognition methods was dominated by handcrafted algorithms based on several image-processing steps. These methods require much effort to design and produce features that potentially differentiate between an object and its local background. The handcrafted features in those methods for any OCR system were obviously language-dependent. It was proven to be a disadvantage in the sense that one algorithm cannot be straightforwardly used to recognize the plate from different countries, as it uses different patterns of characters and different plate formats. Thus, for each different license plate, it requires

*Correspondence to: israa.lewaa@bue.edu.eg; israalewaa@feps.edu.eg

ISSN 2310-5070 (online) ISSN 2311-004X (print)

a significant time to redesign the system. This weakness became the main consideration to develop a modern approach in the recognition system [2]. Arabic characters are essentially cursive and have a large quantity of similarities and few diacritics. These properties make it tough to recognize. A lot of the current strategies shape the detected plate characters with a created template. Independent character recognition is employed to get the characters in form. Hypothesis is that a character of a recognized plate acquiring a major difference in writing model to the one in the template may mistakenly be rejected. A feasibility study is carried out in [20] for a partial OCR on car plate images in Jordan. The place individual character segmentation and recognition are carried out on a limited character set. The segmentation is done using vertical edge detection and then clustering similar regions together. Several connected element analysis is then utilized to determine the location and shape of the characters. Although it is not using Arabic characters, the technique of detecting characters in the plate using the related method may be useful [3]. Creating this system with a high level of accuracy and large variations in Arabic character text recognition will make it a possibility to later invent and implement a system that can recognize Arabic text on various types of media, such as newspapers, street signs and even product packaging. This will enable a high level of accessibility for Arabs living in foreign countries to read Arabic text in the form of translations, without having to depend on human translators to verbally translate for them [4]. In recent years, vehicle license plate recognition has become increasingly popular, driven by its wide range of applications in security, traffic management, and law enforcement. It plays a pivotal role in automatic monitoring, ensuring compliance with regulations on public roads, and controlling access to parking facilities. Additionally, VLPR finds utility in regulating the crossing of borders and enforcing vehicle speed limits. The recognition of a license plate involves several steps, including image acquisition, license plate (LP) extraction, segmentation, and recognition. Image acquisition serves as the initial stage in a VLPR system [1]. A license plate is a rectangular metal plate affixed to a vehicle, featuring numbers, characters, and words that serve as a unique identifier. Since the license plate appears as a distinct region of intensity, an effective identifying method is required to extract the captured images for license plate recognition. Consequently, various methods of deep learning approaches have been employed to identify vehicle license plates, which are explored in our paper [5, 6, 7]. The literature contains several studies focusing on license plate (LP) recognition systems, considering the diverse characteristics of LPs which vary from country to country. The Latin and Arabic alphabets used on LPs in Arab nations are especially distinctive, hence thorough research on the identification processes for these plates is required [3]. However, there have been limited studies employing deep learning methods specifically for Arab nations' license plate detection [8, 9]. Ealier works have also overlooked the inclusion of All governorates' worth of Iraqi plates. This omission can be attributed to the wide range of Iraqi plate types and variations in background color, including white, red, or yellow. To address this gap, this paper considers both Iraqi and Malaysian LPs. While many researches conducted on LPs in Malysia [10, 11, 12, 13, 14, 15], there is a scarcity of research on Iraqi LPs utilizing conventional machine learning method. Furthermore, no prior research has explored the use of deep learning techniques for LP recognition across all regions of Iraq. It is worth mentioning that a recent study has focused on LP recognition in the northern part of Iraq exclusively. In this study, license plate numbers based on Arabic and Latin were used to evaluate the validity of the suggested method. Deep learning was employed as a means to address this problem, providing a solution that was compared against the outcomes of two traditional machine learning techniques. The approah presented in this study is capable of handling license plates with either Arabic or Latin alphabets. The deep learning, unlike traditional methods, eliminates the need for manually engineered features as inputs, instead employing an end-to-end paradigm that automatically learns to segment numbers from the license plate. The following description outlines the general methodology employed in this study: Firstly, the recognition system is implemented using Python. This involves a series of steps. The initial step focuses on identifying the license plate (LP). Subsequently, the license plate is segmented, meaning that the letters, numbers, and words within the LP area are separated. Finally, a deep learning technique is employed to contrast the results obtained from the license plates with the outcomes generated by another model. This paper is organized as follows: In section 1, we present the paper's introduction. A review of the related work for license plate recognition and vehicle identification is presented in Section 2. Section 3 discusses the methodology and simulation. In addition, Section 4 discusses the experiments and results of the study. Finally, the conclusion is presented in section 5.

2. Related Works

Since license plate recognition and vehicle identification are the foundational technologies of intelligent transportation systems, they have attracted a lot of attention in recent years. To do this, several studies in the literature have employed deep learning-based methods. Wang et al. [16] provided a thorough analysis of several DL-based vehicle re-identification methods. These methods were categorized by the authors into five group. Wang et al. investigated the difficulties and future directions for vehicle re-identification study while contrasting various methods. An evaluation of deep learning based models for automatic vehicles that use vision identification (VAVR) was carried out by Boukerche and Ma [17]. They outlined the key issues and current research directions, provided an overview of the various vehicle identification datasets used in VAVR, and enumerated the features of VAVR techniques. A VAVR method was presented by Llorca et al. [18] and was based on modelling the look and shape of vehicle insignia using rearview photos. They employed a linear SVM classifier using HOG features. On a tiny sample of 1.342 images with 52 distinct car models, they achieved an accuracy of 93.75%. Conversely, Shashirangana et al. [19] investigated automatic license plate recognition (ALPR) techniques. They discussed current methods and listed the unresolved issues that the research and development community is now facing. The deep leaning techniques employed in ALPR were divided into two categories by Shashirangana et al. [19], which are single-stage and multi-stage object identification DL-based techniques. Current research has been done on the identification of vehicles and the recognition of license plates. "PROVID," a deep learning method for progressive vehicle re-identification, was presented by Liu et al. [20]. Two progressive search procedures were used by the writers. A CNN model is employed to extract appearance features using a coarse-to-fine search, while a near-to-distant search is utilized for Siamese neural network-based license plate verification. Liu et al. [20] created the VeRi-776 dataset using urban surveillance footage. Three processes make up the DL-based LP detection and identification system that Selmi et al. [21] developed detection, segmentation, and character recognition. Before implementing the three earlier phases, a number of morphological operations are conducted, including geometric filtering, adaptive thresholding, and fine contours. Plate/non-plate classes are used to categorize the identified items in LP detection, which is founded on the CNN model. Then, another CNN model with 37 classes is used to recognize numerals 0–9 and characters in upper case format (A–Z). The issue of multilingual LP detection and recognition was the main focus of the research of Kessentini et al. [22]. Using YOLO proposed by Redmon and Farhadi [23], Kessentini et al. [22] suggested a technique to make the prediction based on two rounds of deep learning: (1) LP recognition on cropped photos and (2) LP detection on raw images using YOLOv2. They conducted a comparison between two recognition engines: a combined recognition strategy at the plate component level and a convolutional recurrent neural network for completing LP recognition without previous segmentation. The Tunisian LP setting was utilized to apply the Kessentini et al. technique. Using YOLO created by Redmon et al. [24], Hendry et al. [25] tackled the issue of automobile LP detection. They modified the original YOLO to create a 36-model single-class detector.

In 2003, Sarfraz et al. [26] presented the first study on the subject of Saudi Licence Plate automatic recognition. They created a dataset with 610 cars in various lighting scenarios. First, vertical edge detection, matching, and filtering are used to find the Saudi Licence Plate. Next, each character is separated by extracting the characters by counting a number of black pixels in each pixel column. Finally, template matching is used to recognize each character once it has been normalized. The Hamming distance is used to associate a character with a class. The technique is now seriously out of date. Furthermore, the format of the license plate that is being utilized does not correspond to the present design of Saudi Licence Plate. Zidouri et al. [27] suggested an additional pipeline. The Sobel filter's intermediate uses vertical edge detection to find the SLP. A thickening mask and a set of predetermined parameters about the SLP position in the picture are then added to improve the detection. Pixel counting is the foundation for character segmentation in each column. Ultimately, multilayer perceptrons in neural networks are employed to identify the segmented characters. Using 61 plates, they evaluated their procedure and were able to identify 97% of them. Recently, several new studies have concentrated on SLP with DL-based methods. A method for automated LP identification based on DL in an unrestricted environment was described by Khan et al. [28]. The authors used a cell phone to record actual traffic footage to create their collection of still photos. The CNN model is utilized to identify the detected alphanumerics, whereas YOLOv5 is used to detect the LP. Khan et al. [28] only

Table 1. Number of images in the training and validation datasets

Item	Training Set	Validation Set	Testing Set	Total
Cairo	1605	153	76	1834

concentrated on English text and ignored Arabic letters in the LP to adapt their approach to other datasets. As a result, they did not utilize all of the data included in Saudi Licence Plates. Approach based on deep learning for the detection and identification of Saudi Licence Plate that was presented by Driss et al. [29]. They created a CNN for LP identification. 1150 images of Saudi cars taken in a variety of lighting and weather situations were used for the experiments. The accuracy of the suggested approach was 98% for LP identification and 92% for LP detection.

3. Materials and Methods

We proposed an approche that utltzied YOLO model which introduced by Redmon et al. in [24]in 2016. In thier study, they introduce a unique methodolgy known as the Darknet, which serves as the basis for a series of the greatest betworks based on real time object detection available today, including YOLOv2 [23], YOLOv3 [30], YOLOv4 [31], YOLOv5, YOLOv7 [32], and its most recent iteration, YOLOv8 [24]. Interestingly, Region-Based Convolutional Neural Networks (R-CNN) were used for object recognition prior to the development of YOLO models. CNNs with architectures like Darknet or FPN (Feature Pyramid Network) are used to extract features from the input image. The detection head receives the output from this layer and uses it to predict the class probabilities and coordinates of the objects in the picture. The model's overall accuracy is increased. After the detection head has predicted the bounding boxes and class probabilities, identical detections are removed using a non-maximum suppression (NMS) technique. The final output of the YOLO model consists of a bounding box set, class probabilities, and objectness scores for each identified object in the image. The detecting head, backbone network, and additional innovations that increase object detection speed and accuracy—like anchor boxes—have all contributed to the evolution of the YOLO models.

3.1. Car and License Plate Detection

Identifying vehicles and license plates in every frame is the first step. In order to achieve this goal, a dataset of 1834 images of cars with license plates in Iraqi, which is available in the study proposed by Mohammed and Abdulsada [33], then manually labeling them. Often, when a car is photographed from the side, its license plate is obscured by other vehicles or other objects. 10% is set aside for validation and 90% is used for training in this dataset. The number of images in the training, validation, and testing datasets is displayed in Table 1.

Using this dataset, A YOLOv8 model was trained with an input size of 640. YOLOv8 was selected because, when in contrast to alternative object detectors like YOLOv3 by Redmon and Farhadi [23], Faster R-CNN by Ren [34], or EfficientDet by Ten et al. [35], it offers a decent trade-off between precision and inference time. We employed 20 pictures for the batch size, a learning rate of 0.000333, and transfer learning from the COCO dataset based on Lin et al. [36]. A sample of car plates is shown in Figure 1.

3.2. Car Model Recognition

A dataset of 1834 images was created to categorize car models. graphs were gathered from the Internet and mobile phone shots were included, and the labels were manually added using the make-model-generation pattern. Three sets of this dataset were created: training, validation, and testing. Since we are using somewhat big datasets, we decided to employ a single random train-validation-test split for the other item detection and classification issue in addition to this car model classification problem challenges that are explained below. However, for small datasets, k-fold cross-validation techniques are better. Furthermore, The image datasets used for testing and validation are just being used to provide a rough estimate of the models' accuracy.

Figure 3.1. Sample of car plate

Table 2. Hyperparameters used for the training of the car model classifier for YOLOv8

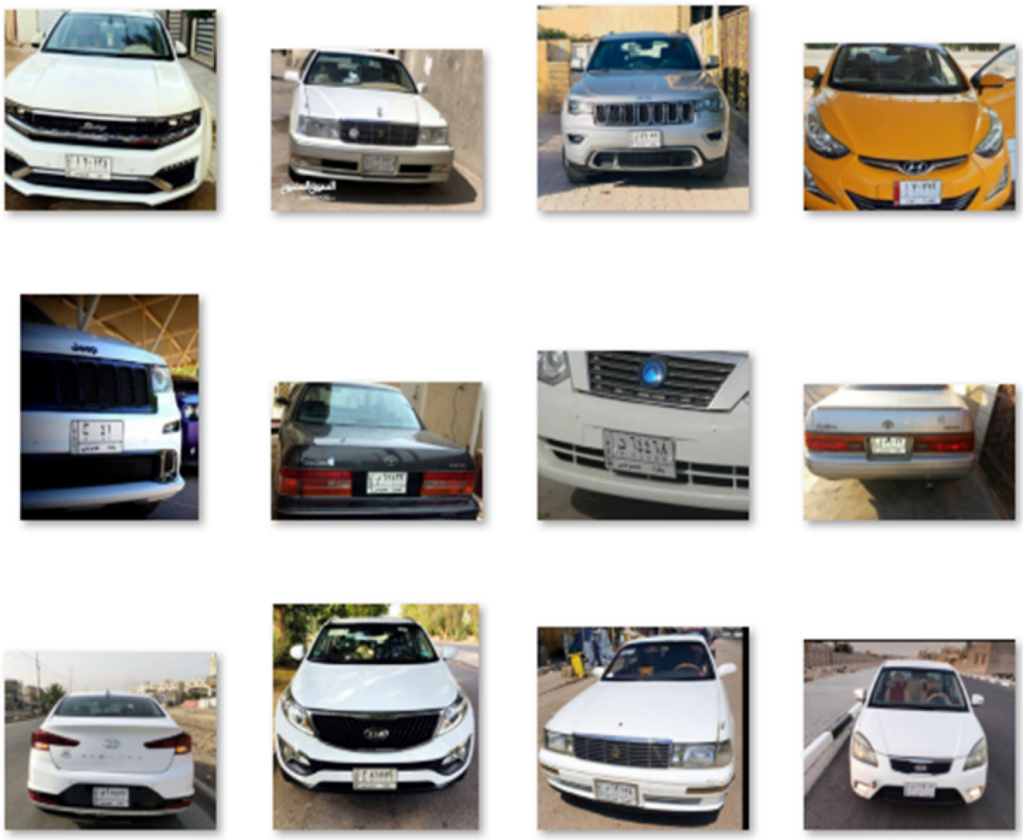
Training Hyperparameter	Value
Image input size	640
Batch size	10
Learning rate	0.000333
Optimizer	Adam
Epochs	75

Figure 2 shows a selection of photos from the dataset used to categorize car models. A collection of 1834 images was created to categorize car models. Images were gathered from the Internet and mobile phone images were added, and the labels were manually added using the make-model-generation pattern. Three sets of this dataset were created: training, validation, and testing. Since we are using relatively large datasets, we decided to use a single random train-validation-test split for the other object detection and classification issues, including the car model classification problem that are detailed below. However, for small datasets, k-fold cross-validation techniques are better. Furthermore, The image datasets used for testing and validation are just being used to provide a rough estimate of the model's accuracy. Table 2 displays hyperparameters that utilized to train the model. Following 75 training epochs, the model achieved 94.4% accuracy and 95.9% precision.

4. Experimental Results

Our paper uses YOLOv5 and YOLOv8, which are mainly based on statistical models. Optimizers are used when running these models to improve the accuracy of the model. Depending on how the optimizer is formulated, it may be thought of as a mathematical function that modifies the network's weights given the gradients and further information. Gradient descent, the greedy method of repeatedly lowering the loss function by following the gradient, is the foundation upon which optimizers are based. These functions might be quite complicated or as basic as deducting the gradients from the weights. In addition to being quicker and more effective than others, better optimizers are also frequently recognized for their ability to generalize well (i.e., with less overfitting). The model's performance can be significantly impacted by the optimizer selection. Through our study stochastic gradient descent(SGD) optimizer is used for YOLOv5, whereas Adam optimizer is used for YOLOv8. Stochastic gradient

Figure 3.2. Sample images of the dataset used for car model classification



descent (SGD) specifically refers to the instance of vanilla GD with a batch size of 1. The most fundamental type of GD is SGD. SGD deducts from the weights the gradient multiplied by the learning rate. Even though SGD is simple, it has solid theoretical underpinnings and is still utilized in edge NN training.

$\theta_i = \theta_i - \alpha \frac{\partial L}{\partial \theta_i}$

Since momentum is theoretically equivalent to gaining velocity, it is sometimes referred to as rolling down a ball. A momentum term, which is computed as the moving average of gradients, modifies the weights. The momentum word γ can be understood as friction or air resistance that causes the momentum to decrease proportionately. Momentum adds an extra hyperparameter but speeds up the training process.

$v_i = \gamma v_i + \alpha \frac{\partial L}{\partial \theta_i}$; where $\theta_i = \theta_i - v_i$

By storing both the weighted average of momentum and the individual learning rate of RMSProp, Adam effectively blends the two concepts. In essence, RMSProp is comparable to momentum. The values of v_i will rise and the learning rate will fall if the gradients are continuously big. This allows the use of higher learning rates and adaptively modifies the learning rate for every parameter. $v_i = \beta v_i + (1-\beta)(\frac{\partial L}{\partial \theta_i})^2$; where $\theta_i = \theta_i - \alpha \frac{\frac{\partial L}{\partial \theta_i}}{\sqrt{v_i + \epsilon}}$

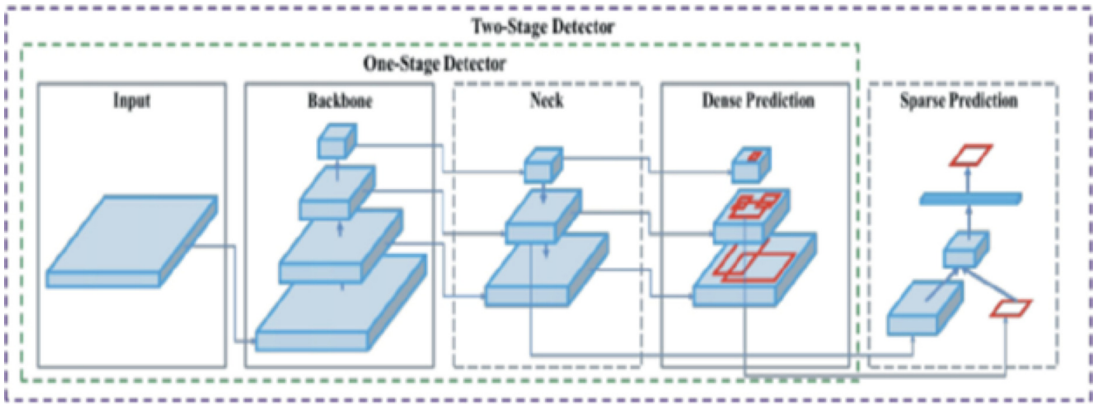
The following formula is used to determine the momentum and RMSProp parameters.

$m_i = \beta_1 m_i + (1-\beta_1)\frac{\partial L}{\partial \theta_i}$

$v_i = \beta_2 v_i + (1-\beta)(\frac{\partial L}{\partial \theta_i})^2$

Before the parameters are applied to the weights in the gradient descent phase, they are split by the (1-decay factor).

Figure 4.1. An illustration of the YOLO model with regard to the head pieces, neck, and backbone



$\hat{m_1} = \frac{m_i}{1-\beta_1}$
$\hat{v_1} = \frac{v_i}{1-\beta_2}$
$\theta_i = \theta_i - \frac{\alpha}{\sqrt{v_l}+\epsilon}\hat{m_l}$

Adam is based on RMSProp, much as in the previous equation, but in order to increase training speed, it estimates the gradient as the momentum parameter. In every training setup and experiment included in the publication, Adam beat every other approach, according to the experiments. Across all sectors, Adam has become the standard optimisation algorithm. Adam, however, adds two more hyperparameters and makes the hyperparameter tuning issue more difficult.

The object recognition system known as YOLOv5 (You Only Look Once version 5) uses a single deep neural network to forecast bounding boxes and class probabilities for items that are present in a picture. The Ultralytics method is an advancement over earlier iterations of the YOLO algorithm, which were also created to aid with real-time object recognition. After being trained on a sizable collection of photos, the algorithm analyses characteristics in the pictures at different sizes to determine which items they are. In order to do this, YOLOv5 extracts attributes via a series of convolutional layers from the image that was entered, and then uses these features to forecast class probabilities and bounding boxes. The input image is normalized and resized to a predetermined size. The images' features are extracted using a backbone network. The backbone of YOLOv5 is the Feature Pyramid Network (FPN). Additionally, the YOLOv5 module consists of a collection of convolutional layers that aid in the fusion of features from various backbone network layers. In YOLOv5, the bounding boxes, objectness scores, and class probabilities are all predicted by the head module. After a series of convolutional layers, it has a set of completely connected layersUsing objectness ratings, class probabilities, and bounding box predictions by the head module, duplicate detections are eliminated and the overall accuracy of the model is increased by the application of the non-maximum suppression (NMS) approach. As seen in Figure 3, YOLOv5 generates a collection of bounding boxes, objectness scores, and class probabilities for each object it detects in the image.

The most recent model in the YOLO family, YOLOv8, is useful for instance segmentation, object recognition, and image classification. The YOLOv5 model's creator, Glenn Jocher, is also behind its development. Since the YOLOv8 model likewise has a FNP made up of head, neck, and backbone components, its operation is essentially the same as that of the YOLOv5. The modules are changed, though. This technique depends on the YOLOv7.

Still, the head module represented the biggest distinction between YOLOv5 and YOLOv8. In YOLOv8, the coupling structure that was initially present in YOLOv5 is replaced by a decoupling one. In addition, the YOLOv8 model does not rely on anchors, but the YOLOv5 model does. A predetermined collection of anchor boxes with different dimensions and aspect ratios is used by an anchor-based model, such as YOLOv5. The bounding boxes' size and placement in relation to these anchor boxes are predicted by the model. Next, the offset between the ground-truth boxes and the anchor boxes is used to modify the predicted bounding boxes. This technique aids

Figure 4.2. A licence plate example using single-character detection

the model in correctly identifying objects with varying sizes and aspect ratios. Conversely, an anchor-free model (such as YOLOv8) does not make use of anchor boxes. Rather, it forecasts the bounding box's size and centre point with precision. This approach gets rid of the requirement to define anchor boxes by hand and simplifies the model. Nevertheless, YOLOv5 and YOLOv8 carry out comparable tasks in terms of training methodologies. Since YOLOv8 is an anchor-free model, it forecasts an object's centre as opposed to its offset from a fully understood anchor box. This method reduces the quantity of box predictions, and as previously shown, this decrease results in a quicker NMS. YOLOv8's bottleneck architecture is the same as YOLOv5, but the kernel size of the first convolution is now 3×3 instead of 1×1. Features are joined directly in the neck, without requiring the same channel dimensions, which results in a reduction of both the size of the relevant tensors and the overall number of parameters.

We first evaluated our solution on a set of 1834 images collected from Iraq images car. We manually labeled images by Roboflow bounding box for each car license. Roboflow [26] provides a network called Object Detection that offers a simple solution to the problem of training and deploying a model. The actual training is carried out using Google Cloud AutoML. The length of training can change based on the number of images and annotations. It is important to note that Roboflow automatically selects the batch size and epoch number and that training ends when the model reaches an overfitting point, which in some circumstances may make the process less error-prone. The public cannot access the algorithmic details of the network, though, as it is a proprietary product. The dataset contains 765 images and we use the Roboflow tool to increase images by performing processing (Auto-oriented) with a resize of 640x640 and Augmentation of 90-rotate. The brightness is between -15% +15% and finally get dataset generation of 1834. License Plate Recognition is done using license plate recognition approaches by evaluating them on the set of images dataset as shown in Figure 4. The two approaches, YOLOv5 and YOLOv8, use 26 classes. To measure the influence of the optimizer of each approach on the accuracy and inference speed, the authors also used in first model approach, yolov8 with auto Adam optimizer and learning rate of 0.01 and momentum=0.937. The best optimizer, learning rate, and momentum are determined automatically. Then, Adam optimizer makes early stopping after 75 epochs with a learning rate of 0.000333 and momentum=0.9.

In the second approach, yolov5, with Stochastic Gradient Descent (SGD)optimizer and learning rate of 0.01 with parameter groups 57 weight(decay=0.0), 60 weight(decay=0.00046875), 60 bias. Table 3 shows of comparison for the difference between the two models. Due to this comparison table yolov8 is better than yolov5.

Figure 4.3. A detection for the license plate for the test image of YOLOv8

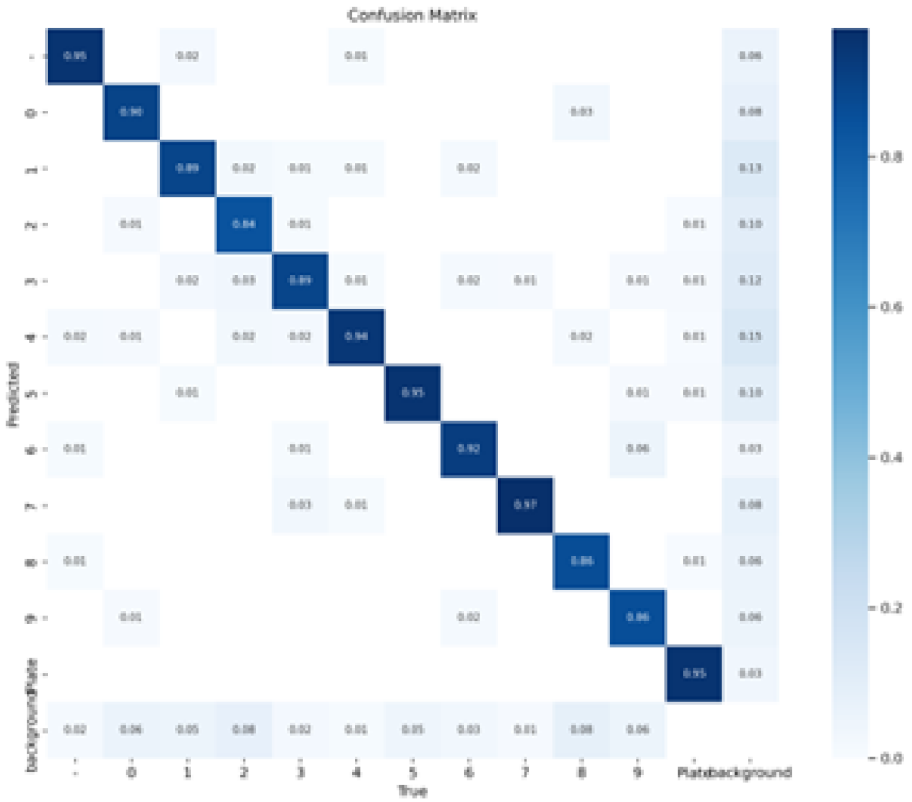
Table 3. Results of Yolov5 and Yolov8

	Yolov5	Yolov8
Accuracy	Precision: 0.95 Recall: 0.905 MAB50: 0.941 MAB50-95: 0.441	Precision: 0.959 Recall: 0.944 MAB50: 0.961 MAB50-95: 0.461
Speed	Training time: 100 epochs completed in 0.811 hours.	Training time: 74 epochs completed in 0.754 hours Speed: 0.3ms preprocess, 2.5ms inference, 0.0ms loss, 5.0ms postprocess per image
Model summary	157 layers, 7080247 parameters, 0 gradients, 16.0 GFLOPs	168 layers, 3010718 parameters, 0 gradients, 8.1 GFLOPs

When evaluating the effectiveness of a classification model, a confusion matrix is commonly employed. It uses the test pictures to compare the predicted class label with the actual class label. Many performance indicators, including F1 score, accuracy, precision, and recall, may be calculated using confusion matrices. The confusion matrix for the YOLOv5 and YOLOv8 models is displayed in Figures 6–7 respectively.

The link between a classifier's confidence level and prediction precision is represented graphically by a precision-confidence curve. Visualizing the trade-off between a model's number of accurate predictions and number of inaccurate predictions at varying confidence levels is a common use in object recognition and classification tasks. The model's prediction quality at various confidence levels is represented by the curve. Although the classifier may be missing some real positive predictions, a high precision at a given threshold suggests that it is producing comparatively few false positive predictions at that confidence level. On the other hand, a poor accuracy at a given threshold suggests that although the classifier may be identifying more real positive predictions, it is simultaneously

Figure 4.4. YOLOv5 model Confusion matrix



producing more false positive predictions. Figures 8-10 suggest that YOLOv8 is somewhat more accurate than the YOLOv5 model at the same confidence level.

5. Conclusion and future research

In this study, we developed and statistically evaluated a multi-stage approach for vehicle identification and license plate recognition tailored to the Arabic context. Extensive testing on the Iraqi License Plate dataset demonstrates reliable performance in controlled settings, with YOLO-based models achieving high accuracy metrics. This system capitalizes on the advancements made in deep learning based object detectors and classifiers. To train these models, the Iraqi License Plate dataset was utilized. One notable component of the system is a customized multi-object tracker, which not only enhances recognition accuracy but also ensures that each car's information is saved only once. The efficiency and effectiveness of the multi-stage framework have been demonstrated through extensive testing under realistic conditions. The results highlight the system's ability to deliver reliable performance in the Arabian context, showcasing its practicality and suitability for real-world deployment. However, there remains a significant disparity between the accuracy achieved on validation datasets in controlled environments and the accuracy attained in real-world unconstrained settings. This discrepancy highlights a significant drawback seen in a significant amount of the body of current work, which frequently only reports findings under restrictive circumstances. Improving the detection, classification, and tracking models through training on bigger and more realistic datasets is essential to closing this gap. Overall, this work emphasizes both the practical application of YOLO-based models and a quantitative, statistical evaluation of their performance,

Figure 4.5. YOLOv8 model Confusion matrix

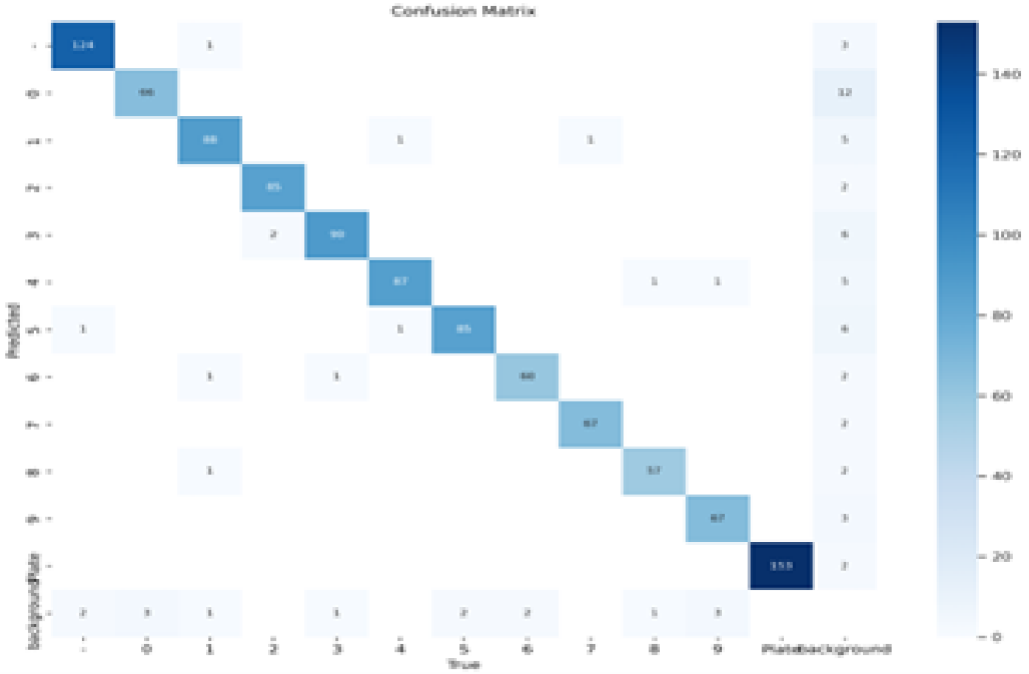
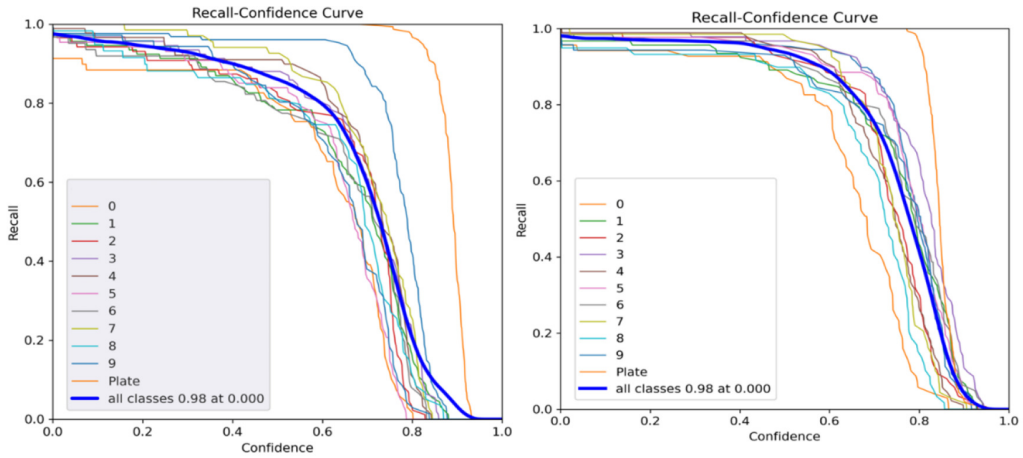


Figure 4.6. YOLOv5 and YOLOv8 Recall Confidence curve



illustrating the system's potential for real-world deployment while providing actionable insights for future model improvements across Arabic and Latin license plate recognition tasks.

Data Availability Statement:

The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy issues.

Conflicts of Interest:

Figure 4.7. YOLOv5 and YOLOv8 Precision Confidence curve

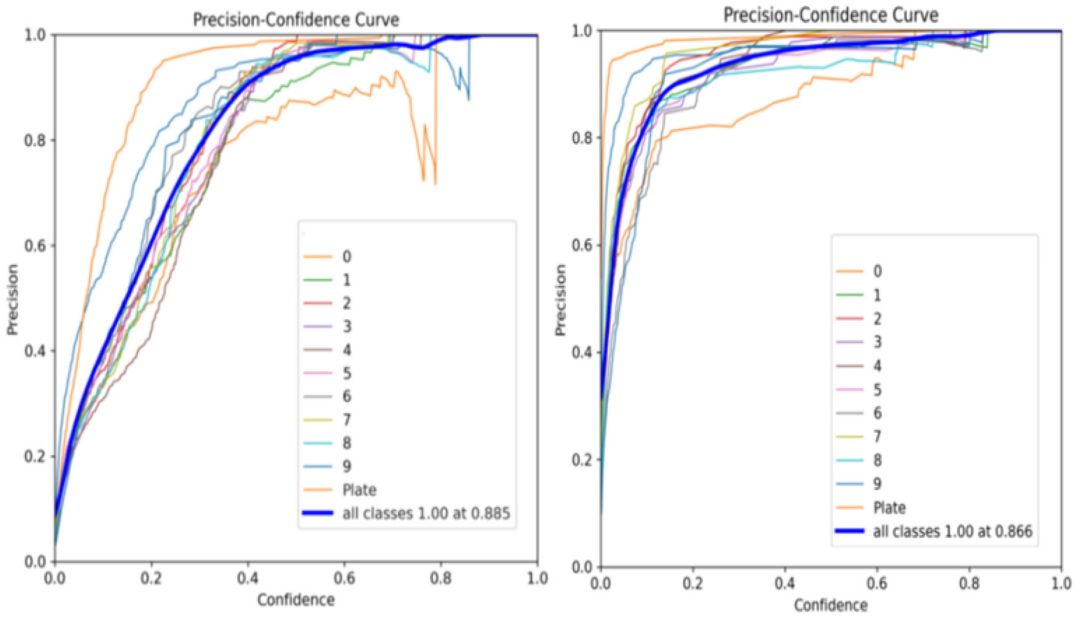
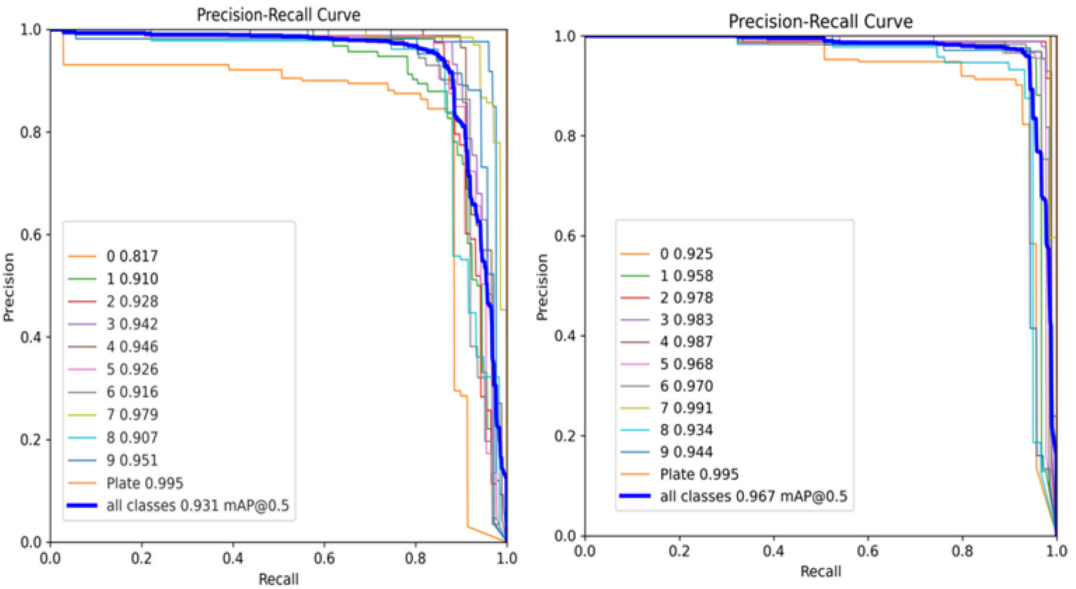


Figure 4.8. YOLOv5 and YOLOv8 Precision-Recall curve



The authors declare no conflict of interest.

REFERENCES

1. Sarfraz, M., Ahmed, M. J., and Ghazi, S. A. (2003). Saudi Arabian license plate recognition system. In *Proceedings of International Conference on Geometric Modeling and Graphics*, 36–41.
2. Omran, S. S. and Jarallah, J. A. (2017). Iraqi car license plate recognition using OCR. In *Proceedings of the 2nd IEEE Annual Conference on New Trends in Information & Communications Technology Applications*, 298–303.
3. Barnouti, N. H., Naser, M. A. S., and Al-Dabbagh, S. S. M. (2017). Automatic Iraqi license plate recognition system using BPNN. In *Proceedings of NTICT*, 105–110.
4. Zhang, Q. et al. (2018). Recent advances in convolutional neural network acceleration. *Neurocomputing*, **77**, 354–377.
5. Jagtap, J. and Holambe, S. (2018). Multi-style license plate recognition using ANN for Indian vehicles. In *ICICET*, 1–4.
6. Borah, P. and Gupta, D. (2017). Support vector machines in pattern recognition. *International Journal of Engineering & Technology*, **9**(3S), 43–48.
7. How, D. N. T. and Sahari, K. S. M. (2017). Character recognition of Malaysian license plates using CNN. In *International Symposium on Robotics and Intelligent Sensors*, 1–5.
8. Kessentini, Y. et al. (2019). A two-stage deep neural network for license plate detection and recognition. *Expert Systems with Applications*, **136**, 159–170.
9. Omar, N. et al. (2020). Cascaded deep learning-based approach for license plate detection. *Expert Systems with Applications*, **149**.
10. Connie, L. et al. (2018). Review of license plate recognition in mobile platforms. *Journal of Telecommunication, Electronic and Computer Engineering*, **10**(3-2), 77–82.
11. Yaacob, N. L. et al. (2021). License plate recognition for campus auto-gate system. *Indonesian Journal of Electrical Engineering*, **21**(1), 128–136.
12. Saleem, N. et al. (2016). Automatic license plate recognition using extracted features. In *IEEE ISCBI*, 221–225.
13. Babu, K. M. and Raghunadh, M. V. (2016). Vehicle number plate detection using bounding box method. In *IEEE ICACCCT*, 106–110.
14. Choong, Y. J. et al. (2020). License plate detection using simplified linear model. *Journal of Critical Reviews*, **7**(3), 55–60.
15. Omar, N. et al. (2020). Cascaded deep learning-based approach. *Expert Systems with Applications*, **149**, 20–25.
16. Wang, H., Hou, J., and Chen, N. (2019). Survey of vehicle re-identification. *IEEE Access*, **7**, 172443–172469.
17. Boukerche, A. and Ma, X. (2021). Vision-based autonomous vehicle recognition. *ACM Computing Surveys*, **54**, 1–37.
18. Llorca, D. F. et al. (2014). Vehicle model recognition using geometry. In *IEEE ITSC*, 3094–3099.
19. Shashirangana, J. et al. (2020). Automated license plate recognition survey. *IEEE Access*, **9**, 11203–11225.
20. Liu, X. et al. (2016). Deep learning-based vehicle re-identification. In *ECCV*, 869–884.
21. Selmi, Z. et al. (2017). Deep learning system for license plate recognition. In *ICDAR*, 1132–1138.
22. Kessentini, Y. et al. (2019). Two-stage deep neural network. *Expert Systems with Applications*, **136**, 159–170.
23. Redmon, J. and Farhadi, A. (2017). YOLO9000. In *CVPR*, 7263–7271.
24. Redmon, J. et al. (2016). YOLO: Unified real-time object detection. In *CVPR*, 779–788.
25. Hendry and Chen, R.-C. (2019). License plate recognition via YOLO. *Image and Vision Computing*, **87**, 47–56.
26. Sarfraz, M. et al. (2003). Saudi Arabian license plate recognition system. In *ICGMG*, 36–41.
27. Zidouri, A. and Deriche, M. (2008). Recognition of Arabic license plates. In *Image Processing Workshops*, 1–4.
28. Khan, I. R. et al. (2022). License plate recognition in real-world traffic videos. *Electronics*, **11**, 1408.
29. Driss, M. et al. (2022). Saudi license plate detection using CNN. In *International Conference on RICT*, 3–15.
30. Redmon, J. and Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
31. Bochkovskiy, A. et al. (2020). YOLOv4. *arXiv preprint arXiv:2004.10934*.
32. Wang, C.-Y. et al. (2022). YOLOv7. *arXiv preprint arXiv:2207.02696*.
33. Mohammed, H. M. and Abdulsada, A. I. (2021). Secure multi-keyword similarity search. *Iraqi Journal for Electrical & Electronic Engineering*, **17**(2).
34. Ren, S. et al. (2015). Faster R-CNN. In *NeurIPS*, Volume 28.
35. Tan, M. et al. (2020). EfficientDet. In *CVPR*, 10781–10790.
36. Lin, T. Y. et al. (2014). Microsoft COCO dataset. In *ECCV*, 740–755.