Hindawi
Journal of Sensors
Volume 2024, Article ID 4917097, 18 pages
https://doi.org/10.1155/2024/4917097

Research Article

Efficient Multistage License Plate Detection and Recognition Using YOLOv8 and CNN for Smart Parking Systems

Mejdl Safran, Abdulmalik Alajmi, and Sultan Alfarhood

Department of Computer Science, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia

Correspondence should be addressed to Mejdl Safran; mejdl@ksu.edu.sa

Received 2 September 2023; Revised 8 January 2024; Accepted 10 January 2024; Published 8 February 2024

Academic Editor: Stanislav Vítek

Smart parking systems play a vital role in enhancing the efficiency and sustainability of smart cities. However, most existing systems depend on sensors to monitor the occupancy of parking spaces, which entail high installation and maintenance costs and limited functionality in tracking vehicle movement within the car park. To address these challenges, we propose a multistage learning-based approach that leverages existing surveillance cameras within the car park and a self-collected dataset of Saudi license plates. The approach combines YOLOv5 for license plate detection, YOLOv8 for character detection, and a new convolutional neural network architecture for improved character recognition. We show that our approach outperforms the single-stage approach, achieving an overall accuracy of 96.1% versus 83.9% of the single-stage approach. The approach is also integrated into a web-based dashboard for real-time visualization and statistical analysis of car park occupancy and vehicle movement with an acceptable time efficiency. Our work demonstrates how existing technology can be leveraged to improve the efficiency and sustainability of smart cities.

1. Introduction

Smart cities are urban areas that use various technologies and data sources to improve the quality of life and efficiency of services for their citizens. One of the key aspects of smart cities is smart transportation, which optimizes mobility and safety in urban environments. Smart parking systems (SPSs) are a crucial component of smart transportation, as they can reduce traffic congestion, fuel consumption, air pollution, and parking search time by providing real-time information on parking availability and location [1]. Rapid advances in communication and information technologies have enabled the development of cost-effective SPSs [2]. These systems benefit car park operators, drivers, and the environment [3]. Operators can analyze car park data to predict future parking patterns and make pricing recommendations. Drivers can find vacant parking spaces and locate their parked cars more easily, reducing travel and search time. SPSs also reduce air pollution by decreasing vehicle emissions [4].

Various SPS approaches have been proposed in the literature. According to Fahim et al. [5], there are 12 different technological approaches to developing SPSs, with wireless sensor networks, the Internet of Things, and computer vision-based systems being the most popular. However, designing and implementing SPSs presents several challenges. One of the main challenges is monitoring parking space occupancy reliably and cost-effectively. Many existing systems use sensors, such as magnetic, ultrasonic, or infrared sensors, to detect vehicle presence or absence in each parking space. However, these systems often have high hardware and maintenance costs and limited capabilities for tracking vehicular movement within the car park. While they can detect whether a parking space is occupied, they do not provide information about the vehicle's identity occupying the space [6]. They may also suffer from sensor failures, interference, or vandalism, affecting their accuracy and reliability [7].

To address these limitations, some recent studies have proposed to use of computer vision techniques to monitor the occupancy of parking spaces using existing surveillance cameras within car parks. These techniques can detect and recognize the license plates of vehicles entering and exiting car parks and update the occupancy status accordingly.

Compared to sensor-based systems, these techniques can reduce hardware and maintenance costs and have more flexibility in tracking vehicle movement within car parks. Furthermore, these techniques can provide additional information and functionality for car park management, such as vehicle identification, access control, security monitoring, and statistical analysis [8].

However, most of the existing computer vision techniques for SPSs are designed for specific regions or countries and may not be applicable or effective for Saudi Arabia, which has different license plate formats, fonts, colors, or backgrounds. Therefore, we propose a novel approach that is tailored for the Saudi license plates and achieves better accuracy and efficiency in detection and recognition. Our approach is based on a multistage learning-based pipeline for SPSs that utilizes existing surveillance cameras within car parks and leverages a self-collected dataset of Saudi license plates. The main contributions of this research are as follows:

(i) We introduce a new, local dataset of Saudi license plates, which we collected and annotated ourselves. This dataset covers various scenarios of license plate detection, character detection, and character recognition.

(ii) We develop a multistage learning-based approach for SPSs that utilizes existing surveillance cameras within the car park and our self-collected dataset of Saudi license plates. Our approach consists of three stages: license plate detection, character detection, and character recognition.

(iii) We train and fine-tune the state-of-the-art YOLO series (i.e., YOLOv5, YOLOv7, and YOLOv8) for robust and fast Saudi license plate detection and character detection. We achieve a mean average precision of 99.4% for license plate detection using YOLOv5x and a mean average precision of 98.1% for character detection using YOLOv8x.

(iv) We propose a new convolutional neural network (CNN) architecture for improved license plate character recognition, which surpasses other existing approaches with an accuracy of 97%.

(v) We demonstrate experimentally that our multistage approach outperforms the single-stage approach (YOLOv8x) with an overall accuracy of 96.1% versus 83.9%, with a reasonable time efficiency.

(vi) We develop a web-based dashboard that enables real-time monitoring and statistical analysis of car park occupancy. This dashboard serves as a practical demonstration of our system in a real-world scenario.

The rest of this paper is organized as follows: In Section 2, we provide a comprehensive review of the existing SPSs that use deep learning techniques. In Section 3, we explain our proposed approach in detail and describe the experiment preparation and setup. In Sections 4 and 5, we present and analyze the experimental results and discuss the limitations and future work of our approach. Finally, in Section 6, we summarize the main contributions and findings of this paper.

2. Related Work

In this section, we review various techniques proposed in the literature for SPSs using license plate detection and recognition (LPDR). We then discuss LPDR approaches applied on Saudi license plates and conclude by highlighting the contributions of our proposed approach compared to existing works.

2.1. Overview of LPDR Approaches for SPSs. Several LPDR approaches have been proposed for SPSs. One approach is to use neural networks for identifying car license plates. For example, Sirithinaphong and Chamnongthai [9] proposed a method for identifying car license plates for an automatic parking system using car license plate patterns and a four-layer neural network with an accuracy of 96% for license plate extraction and 92% for recognition on 70 car images with the automatic parking system prototype. Another approach is to use the YOLO model to find the license plate region and the ResNet model to recognize the characters. Joshua et al. [10] presented such a method for identifying vehicle license plates for a parking system with around 80% accuracy for character recognition on Indonesian license plates. A solution for live detection and recognition of a moving vehicle's license plate number using computer vision techniques was proposed by Darapaneni et al. [11] with the best performance in terms of accuracy and speed using YOLOv3 with OpenCV. Thakur et al. [12] presented a web-based system for parking lot car management that uses computer vision and the YOLOv3 deep learning model for license plate recognition with an accuracy of 99.2% for license plate detection and 96.1% for character recognition. Neupane et al. [13] proposed SHINE, a deep learning-based accessible parking management system that detects vehicles, license plates, and disability badges in real-time using a YOLOv7 model for both object detection and license plate recognition with a mean average precision of 92.16% on a custom dataset of 30,000 images collected from various parking lots in South Korea.

Some researchers have used autocorrelation, mean square error, and structural similarity index for managing and collecting parking fees using vehicle number plate recognition, such as Rashid et al. [14], with a recognition rate of 95.6% using the mean square error approach. Another approach is to use license plate recognition and RFID for car parking management, such as what was presented by Sen et al. [15]. Van et al. [16] presented a ticketless parking system that uses automated license plate recognition (ALPR) and face verification technologies to authenticate the vehicle's owner with high accuracy using various AI models and algorithms, such as 99.2% accuracy for detecting four-wheeler license plates, 96.1% accuracy for recognizing car license plate characters, 95% average precision for detecting masked face features, and 76.67% for verifying face identity.

Thai-Nghe and Chi-Ngon [17] proposed a solution for building an intelligent parking support system that integrates three recognition techniques: automatic recognition for motorcycle license plate, barcode recognition, and semiautomatic recognition via surveillance cameras with an accuracy of 99% for license plate area recognition, 95.88% for letter area recognition, and 98.99% for classification.

A web-based system for car management parking lots that uses computer vision and open automatic license plate recognition (OpenALPR) technologies was presented by Shkurti et al. [18] with the ability to recognize 85% of images captured from the camera. Thai et al. [19] developed a system that can predict parking availability based on a long short-term memory (LSTM) network and notify the drivers about forecast information. Similarly, Dalarmelina et al. [20] used Tesseract OCR, which implements an LSTM network, to detect parking slot availability by recognizing license plate characters of vehicles moving around or parked in the parking slots, achieving an accuracy of 83%. Xiang and Pan [21] proposed a system that uses sensors, wireless network, controller, and an improved Faster R-CNN deep learning algorithm to monitor and manage parking spaces with accuracies of 92.84%, 90.19%, and 85.31% in three different difficulty levels. Anuar and Lingas [22] presented an SPS that focuses on car entrance, exit, and parking management using sensors and technologies to increase efficiency and reduce complications, achieving a detection rate of 97.1% and a recognition rate of 48.6%.

2.2. LPDR Approaches Applied on Saudi License Plates. Several LPDR approaches have been applied on Saudi license plates. Some studies have focused on improving existing LPDR techniques to make them more suitable for license plates from Arabic countries such as Egypt, Saudi Arabia, and the UAE. For instance, one study improved the Faster R-CNN framework and achieved a recall of 98.65% and a precision of 97.46% on a dataset of license plates from these countries [23]. Another study applied transfer learning to train the YOLOv5 framework to detect license plates and alphanumerics and trained a CNN to recognize detected alphanumerics with a recognition rate of 92.8% on 2,600 car images collected from real traffic videos in Saudi Arabia [24].

Some approaches have utilized the information redundancy of Saudi license plates' Arabic and English characters to boost the accuracy of license plate recognition while maintaining real-time inference performance. For example, Ammar et al. [25] introduced a multistage, real-time, deep learning-based system for vehicle identification and license plate recognition that achieved detection rates of 81.9% and 80% and recognition rates of 67% and 95% on two videos of vehicles and license plates at several parking entrance gates. Khan et al. [26] presented a deep learning-based license plate recognition system that utilizes bilingual text in license plates to restore noise-affected missing or misidentified characters in the plate with an accuracy of 99.5% for character recognition and 97% for plate detection.

Other studies have proposed automatic license plate recognition systems using various combinations of deep learning techniques such as Faster-RCNN, CNN, YOLO, and radial basis function (RBF) neural networks for license plate detection and character recognition. For example, Driss et al. [27] proposed an automatic license plate recognition system that uses Faster-RCNN for license plate detection and CNN for character recognition with a precision of 92% for license plate detection and an accuracy of 98% for license plate recognition. Maglad [28] proposed a system for detecting and recognizing Saudi Arabian license plates using an RBF neural network for both detection and recognition with an accuracy of 95% for license plate detection, 99% for character segmentation, and 91% for character recognition. Alyahya et al. [29] presented a system for recognizing Saudi license plate numbers using a bi-linear interpolation algorithm, preprocessing techniques, and an OCR based on an ANN classifier with an accuracy of 92%.

Other studies have employed shallow machine learning techniques, such as SVM, RBF, and k-nearest neighbors classifiers, to recognize Saudi license plates. For example, Suwais et al. [30] developed an SVM-based algorithm that achieved 93.3% accuracy. Another study [31] compared the performance of SVM and RBF classifiers and found that the RBF-based method outperformed the SVM-based method. Alzubaidi et al. [32] proposed a Raspberry PI-based system that used pixel distribution, horizontal projection profiles, distance classifier, and k-nearest neighbors classifier to achieve 90.6% accuracy.

2.3. Remarks. Our proposed approach offers several distinct advantages over existing SPSs utilizing LPDR approaches, as well as over existing LPDR approaches applied specifically to Saudi license plates. First, our approach diverges from existing systems that rely on sensors to monitor parking space occupancy by leveraging preexisting surveillance cameras within the car park in conjunction with our self-collected dataset of Saudi license plates. This approach not only reduces installation and maintenance costs but also enhances functionality in tracking vehicle movement within the car park. Second, our approach for license plate detection and recognition addresses the tradeoff between the accuracy and time efficiency of SPSs using state-of-the-art YOLO models and a newly proposed CNN architecture. Finally, we have developed a web-based dashboard for real-time monitoring and statistical analysis of car parks, demonstrating the efficacy of our approach in a real-world scenario experiment. This dashboard provides a practical solution for parking management systems, enabling efficient monitoring and analysis of car park occupancy and vehicle movement.

3. Materials and Methods

We herein detail our methodology. First, we discuss the proposed system architecture. Then, we describe the stages employed in the proposed deep learning approach. We then present the application of our proposed system to a use case in Saudi Arabia. Finally, we describe the experiment preparation and setup, including the dataset, the hardware, and the software. We also discuss the evaluation metrics that we used to measure the performance of our system.

3.1. System Architecture. The proposed system architecture, as illustrated in Figure 1, comprises three main components:

(i) The Car Park Agent consists of car park cameras connected to a local PC equipped with the proposed deep learning model to analyze the cameras' footage by detecting and recognizing car license plates. One camera is positioned at the car park entrance, while

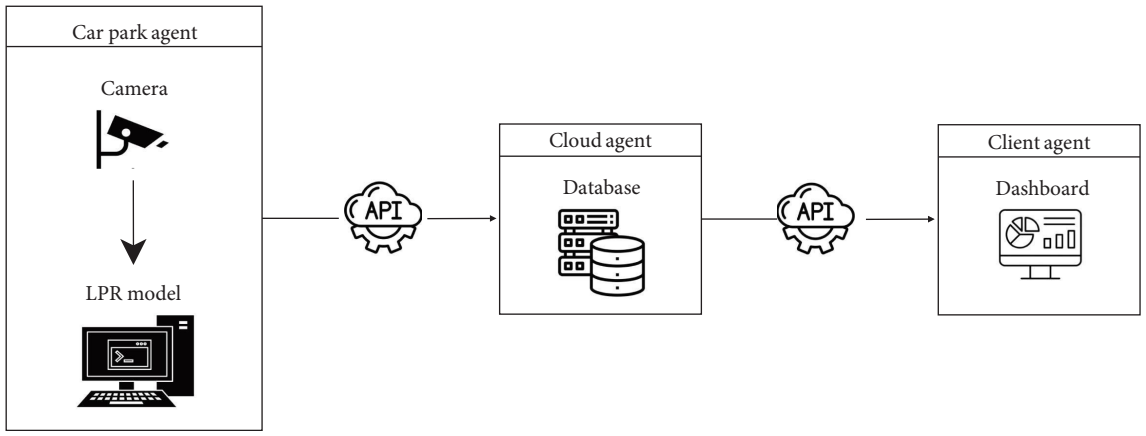


Figure 1: Proposed system architecture.

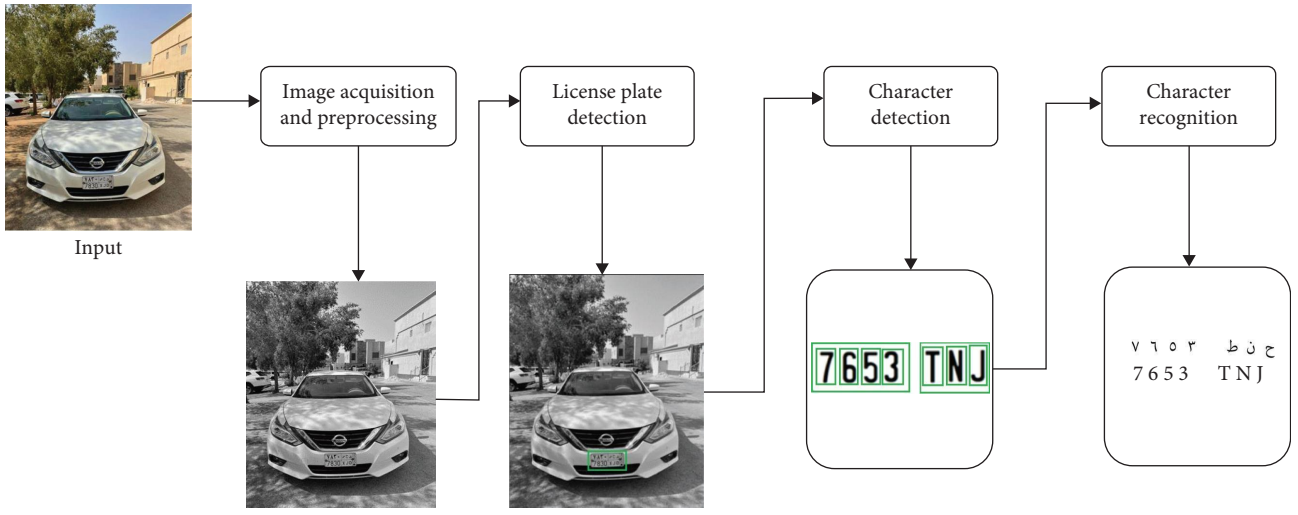


Figure 2: Phases of license plate detection and recognition.

additional cameras are located at the entrance of each floor. The deep learning model leverages the state-of-the-art YOLO series and a newly proposed CNN architecture for improved license plate character recognition, achieving high accuracy and acceptable time efficiency.

(ii) The Cloud Agent is a cloud-based database that receives insights from the Car Park Agent via APIs. To maintain privacy and confidentiality, camera footage is not transmitted to the Cloud Agent. Instead, the Cloud Agent receives insights derived from the Car Park Agent's analysis of the footage, enabling real-time monitoring and analysis of car park occupancy and vehicle movement.

(iii) The Client Agent is a web-based dashboard that retrieves statistical results from the Cloud Agent via APIs. These results are displayed to car park operators, enabling efficient monitoring and analysis of car park occupancy and vehicle movement in real-time.

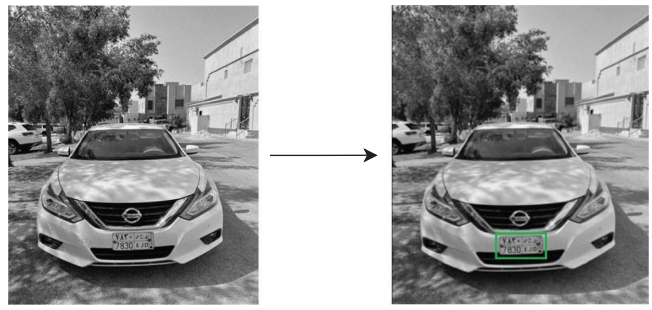
In the following subsection, we will delve into the details of the proposed deep learning approach for license plate detection and recognition, which is deployed in the Car Park Agent. We will discuss the various stages of the model and its implementation in detecting and recognizing Saudi license plates.

3.2. License Plate Detection and Recognition. License plate detection and recognition is a crucial component of the Car Park Agent in our proposed system. We herein describe in detail the steps involved in building the deep learning model for detecting and recognizing Saudi license plates in car parks. As illustrated in Figure 2, the model consists of four phases: (a) image acquisition and preprocessing, (b) license plate detection, (c) character detection, and (d) character recognition. We will discuss each of these phases in detail in the following subsections.

3.2.1. Image Acquisition and Preprocessing. Image acquisition and preprocessing is the first phase in our deep learning model for license plate detection and recognition. Image

Input Output

Figure 3: Preprocessing phase.



Input Output

Figure 4: License plate detection phase.

preprocessing plays a crucial role in the object detection model pipeline by highlighting the desired region in the image. During this phase, various filters are applied to the image to remove impurities and enhance its quality. In our work, images are resized to 640×640 pixels and converted to grayscale using resize and cvtColor methods from the OpenCV-Python library, respectively, as shown in Figure 3. These specific image dimensions were chosen to match the standard dimensions used by YOLO models trained on the COCO dataset [33], enabling the use of transfer learning when training our proposed models in subsequent phases. Images are also denoised using the fastNlMeansDenoising method from the OpenCV-Python library and transformed by increasing their brightness using the ColorJitter (brightness = 0.2) method provided by PyTorch. Moreover, we used augmentation techniques on the training set images when training our YOLO and CNN models on our dataset. These techniques helped us to simulate real-world scenarios in parking systems and improve the generalization of our models. The validation and test sets were not augmented to ensure an accurate evaluation of the proposed model performance. We will discuss the augmentation techniques when we present the training process of the proposed models.

3.2.2. License Plate Detection. License plate detection is the second phase in our deep learning model for license plate detection and recognition. The primary objective of this phase is to identify and extract the license plate from an image of a car. This is achieved by searching the car image for the rectangle that contains the license plate. In this phase, we train and fine-tune YOLOv5x [34], YOLOv7x [35], and YOLOv8x [36] on a self-collected dataset of Saudi license plates, which will be discussed in detail later in this section. The model receives a car image as input and outputs the detected license plate, as illustrated in Figure 4.

YOLO object detection algorithms are based on the single-shot detection framework, which makes object detection by dividing the input image into grids and predicting bounding boxes and confidence scores for each grid cell. YOLOv5 [34] uses a cross-stage partial network (CSPNet) [37] and Darknet as a backbone, which is responsible for extracting features from the input image. The neck of YOLOv5 is a path aggregation network (PANet) [38], which fuzes features from different levels of the backbone. YOLOv5 also uses adaptive feature pooling to improve the accuracy of object location. YOLOv5 has five different model scales that can gain a tradeoff between size and performance: YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. YOLOv7 [35] introduces the extended efficient layer aggregation network (E-ELAN) [39] as a better version of the ELAN computational block, which enables efficient learning without gradient loss. YOLOv7 uses reparameterized convolutions (RepConv) [40] as the basic building block and applies coarse label assignment for the auxiliary head and acceptable label assignment for the lead head. YOLOv7 has two model sizes: YOLOv7 and YOLOv7x. YOLOv8 [36] is an anchor-free detector that reduces the number of box predictions and speeds up the non-maximum suppression, which is a process of removing overlapping bounding boxes. YOLOv8 employs mosaic augmentation to enhance the training process for various real-world applications. YOLOv8 is considered the state-of-the-art among YOLO object detectors [41]. YOLOv8 has five model scales: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. Table 1 shows the specifications of all versions of YOLOv5, YOLOv7, and YOLOv8.

Table 1: Specifications of all versions of YOLOv5, YOLOv7, and YOLOv8.

Model	Parameters (millions)	Year	Reference
YOLOv5n	1.8	2020	[34]
YOLOv5s	7.1		
YOLOv5m	20.9		
YOLOv5l	46.1		
YOLOv5x	86.2		
YOLOv7	37.2	2022	[35]
YOLOv7x	70.8		
YOLOv8n	3.0	2023	[36]
YOLOv8s	11.2		
YOLOv8m	25.9		
YOLOv8l	43.6		
YOLOv8x	68.2		

3.2.3. Character Detection. Character detection is the process of decomposing an image of a sequence of characters into subimages of individual symbols. This involves extracting individual letters and digits from an image, such as a detected license plate.

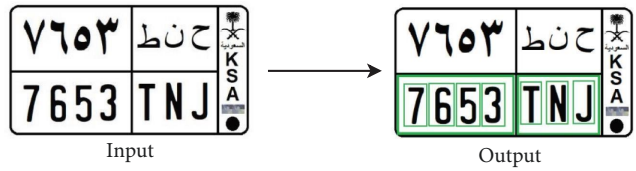


Figure 5: Character detection phase.

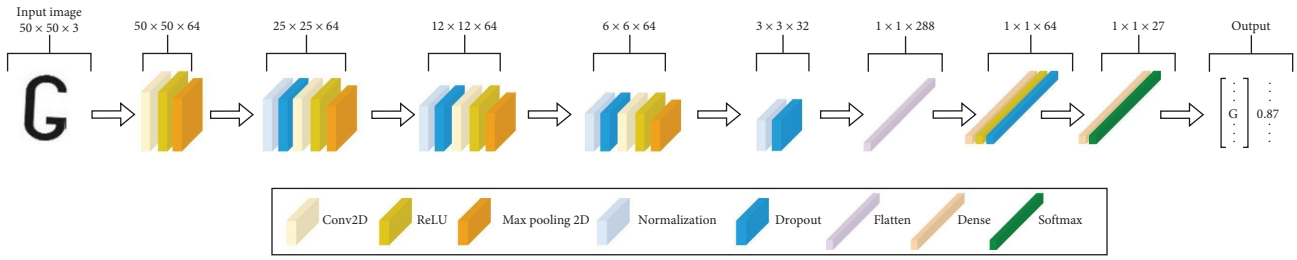


Figure 6: Character recognition model architecture.

We use adaptive thresholding to convert the detected license plate to a binary image, which improves the detection process. Adaptive thresholding is a technique that adjusts the threshold value based on the local pixel neighborhood, resulting in a more robust binarization. The OpenCV library provides the cv2.adaptiveThreshold function for this purpose. This function takes four parameters: ADAPTIVE_THRESH_MEAN_C, THRESH_BINARY, 15, and 20. The first parameter specifies that the function will use the mean of the neighborhood area as a basis for the threshold value. The second parameter specifies that the function will compare each pixel value with the threshold value and assign it either 0 or 255, depending on whether it is below or above the threshold. The third parameter specifies the size of the pixel neighborhood, which is a square of 15×15 pixels. The fourth parameter specifies a constant value that the function will subtract from the mean to fine-tune the threshold value.

Unnecessary characters, such as Arabic characters, Hindi digits, plate boundaries, and constant characters (e.g., the word KSA), are then eliminated. Two large bounding boxes are extracted, one for the English letters and one for the Arabic digits. Within these large bounding boxes, smaller bounding boxes are extracted for each individual letter and digit. To train models for character detection, a new dataset is generated by manually annotating the regions of interest in the detected license plates, which will be discussed in detail later in this section. We train and fine-tune YOLOv5x, YOLOv7x, and YOLOv8x to detect the bounding boxes of the English letters and Arabic digits. The trained models then detect each letter and digit within each bounding box.

The model receives the detected license plate as input and outputs localized English letters and Arabic digits (the regions of interest). Figure 5 shows an example of the input and output of the character detection phase.

3.2.4. Character Recognition. Character recognition is the process of recognizing detected characters and mapping them to their corresponding Arabic letters and Hindi digits. In this phase, a simple CNN framework is built and trained for character recognition. The proposed CNN framework is illustrated in Figure 6, which consists of four convolutional layers and two dense layers. The first convolutional layer has 64 filters of size 4×4 and applies the same padding and ReLU activation to the input images of shape $50 \times 50 \times 3$. The output of this layer is then passed through a max pooling layer with default parameters, a normalization layer that scales the inputs to have zero mean and unit variance, and a dropout layer that randomly sets 25% of the inputs to zero to prevent overfitting. The second and third convolutional layers have the same structure as the first one. The fourth convolutional layer has 32 filters of size 2×2 and also applies the same padding, ReLU activation, max pooling, normalization, and dropout as the previous layers. The output of the fourth convolutional layer is then flattened into a 1D vector and fed into a dense layer with 64 units and ReLU activation. Another dropout layer with a 50% rate is applied before the final dense layer with 27 units and softmax activation, which produces the probability distribution over the 27 classes. The model has a total of 163,263 parameters, of which 162,811 are trainable and 452 are nontrainable. The nontrainable parameters are from the normalization layers, which have fixed statistics for scaling the inputs.

The model receives a detected license plate image as input and attempts to recognize the plate characters. As a final step, English letters are mapped to Arabic letters, and Arabic digits are mapped to Hindi digits, as shown in Figure 7. An example of the input and output of the character recognition phase is presented in Figure 8.

Our approach consists of detecting and recognizing only English letters and Arabic digits from the license plates and then mapping them to their corresponding Arabic letters and Hindi digits. We have adopted this strategy for three main reasons: (a) previous studies have demonstrated that using only English letters and Arabic digits can achieve high accuracy, as evidenced by Khan et al. [24] and Ammar et al. [25]; (b) some Arabic letters and Hindi digits have similar or

Arabic letter	English letter
ا	A
ب	B
ت	
ث	
ج	
ح	J
خ	
د	D
ذ	
ر	R
ز	
س	S
ش	
ص	X

Arabic letter	English letter
ض	
ط	T
ظ	
ع	E
غ	
ف	
ق	G
ك	K
ل	L
م	Z
ن	N
هـ	H
و	U
ي	V

Hindi digit	Arabic digit
٠	0
١	1
٢	2
٣	3
٤	4
٥	5
٦	6
٧	7
٨	8
٩	9

Figure 7: Mapping English letters to Arabic letters and Arabic digits to Hindi digits. Letters that are not used in Saudi license plates are highlighted. Hindi digits, also known as Eastern Arabic numerals or Indic numerals, are derived from the Arabic script and used for writing Arabic and other languages. Arabic digits, also known as Western Arabic numerals or Hindu-Arabic numerals, are the numerals that are widely used in the Western world.

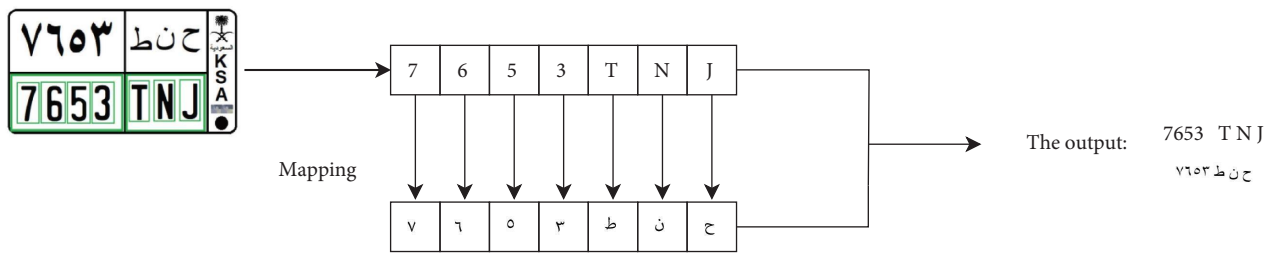


Figure 8: Character recognition phase.

Figure 9: Challenges in direct detection and recognition of Arabic letters and Hindi digits.

ambiguous shapes or orientations, which can lead to confusion or errors in the recognition process. For instance, the Arabic letter Alif (equivalent of English letter A) resembles the Hindi digit one (equivalent of Arabic digit one); and (c) many Saudi license plates are attached to the cars using bolts on top-right and top-left regions which can interfere with the Arabic letters and Hindi digits and make their detection and recognition more challenging. For example, the bolts can cause the Hindi digit one (equivalent of Arabic digit one) to be read as the Hindi digit nine (equivalent of Arabic digit nine), and the Arabic letter Alif (equivalent of English letter A) to be read as Hindi digit nine (equivalent of Arabic digit nine). Figure 9 shows some examples of these cases.

3.3. SPS Dashboard. The Client Agent provides a web-based dashboard for sending and receiving data from the Cloud Agent using APIs, as we discussed and showed in Figure 1. The dashboard is designed to facilitate the management and analysis of car park operations by providing real-time and statistical information about car park occupancy and vehicle movement. The dashboard has a user-friendly interface that allows the operator to customize the display of information according to various criteria, such as date range, car park branch, floor number, and license plate number. The dashboard also supports data visualization using graphs and charts to enhance the understanding of the car park performance and trends. The dashboard reflects the business workflow of the proposed system, which consists of the following steps:

(i) Step 1: A car approaches the entrance gate of the car park. A camera at the gate captures the image of the car and sends it to the local PC, which has the proposed deep-learning models for license plate detection and recognition installed.

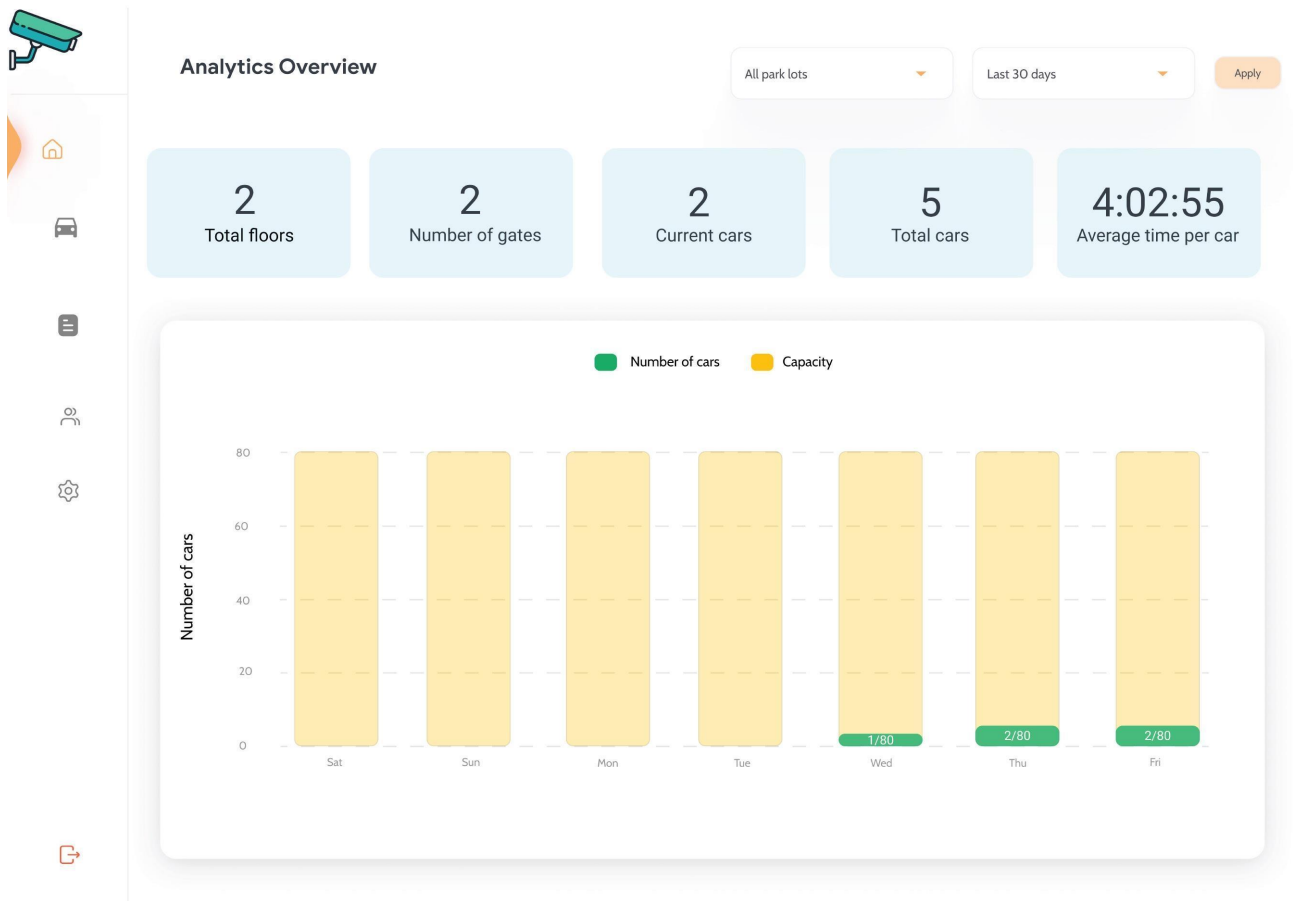


Figure 10: Smart parking system dashboard homepage.

(ii) Step 2: The YOLO models analyze the image and detect the license plate of the car and its characters. The license plate characters are then recognized using the newly proposed CNN architecture. A unique ID is assigned to the car, and the entry date and time are recorded.

(iii) Step 3: The local PC sends the car ID, current floor number, license plate number, and entry date and time to the Cloud Agent via APIs. The Cloud Agent stores this information in a cloud-based database and updates the car park occupancy status. The Client Agent receives a single to update the dashboard with the latest information about car park occupancy and vehicle movement. To avoid multiple detection and recognition of the same car and to reduce the computation power required to process the frames, the model implements a redundancy check mechanism. The mechanism compares the license plate number of the current frame with the license plate numbers of the previous frames stored in a buffer. If the license plate number of the current frame matches any of the license plate numbers in the buffer, the model omits sending and processing the current frame.

(iv) Step 4: If the car moves to another floor/zone, a camera at the entrance or exit of each floor captures the image of the car and sends it to the local PC. The local PC verifies the license plate number of the car and records the new floor number. It then sends this information to the Cloud Agent and the Client Agent via APIs. The Cloud Agent and the Client Agent update their respective databases and dashboards accordingly.

(v) Step 5: When the car is ready to exit the car park, it approaches the exit gate. A camera at the exit gate captures the image of the car and sends it to the local PC. The local PC verifies the license plate number of the car and records the exit date and time, and the dwell time. It then sends this information to the Cloud Agent and the Client Agent via APIs. The Cloud Agent and the Client Agent update their respective databases and dashboards accordingly.

The dashboard offers many features to enhance the efficiency of car park management. Some of its key capabilities are described below:

(i) Homepage: The homepage, shown in Figure 10, displays real-time and statistical information about the car park. The operator can filter the information by date range, car park branch (for multibranch car parks), and floor number. The homepage also shows

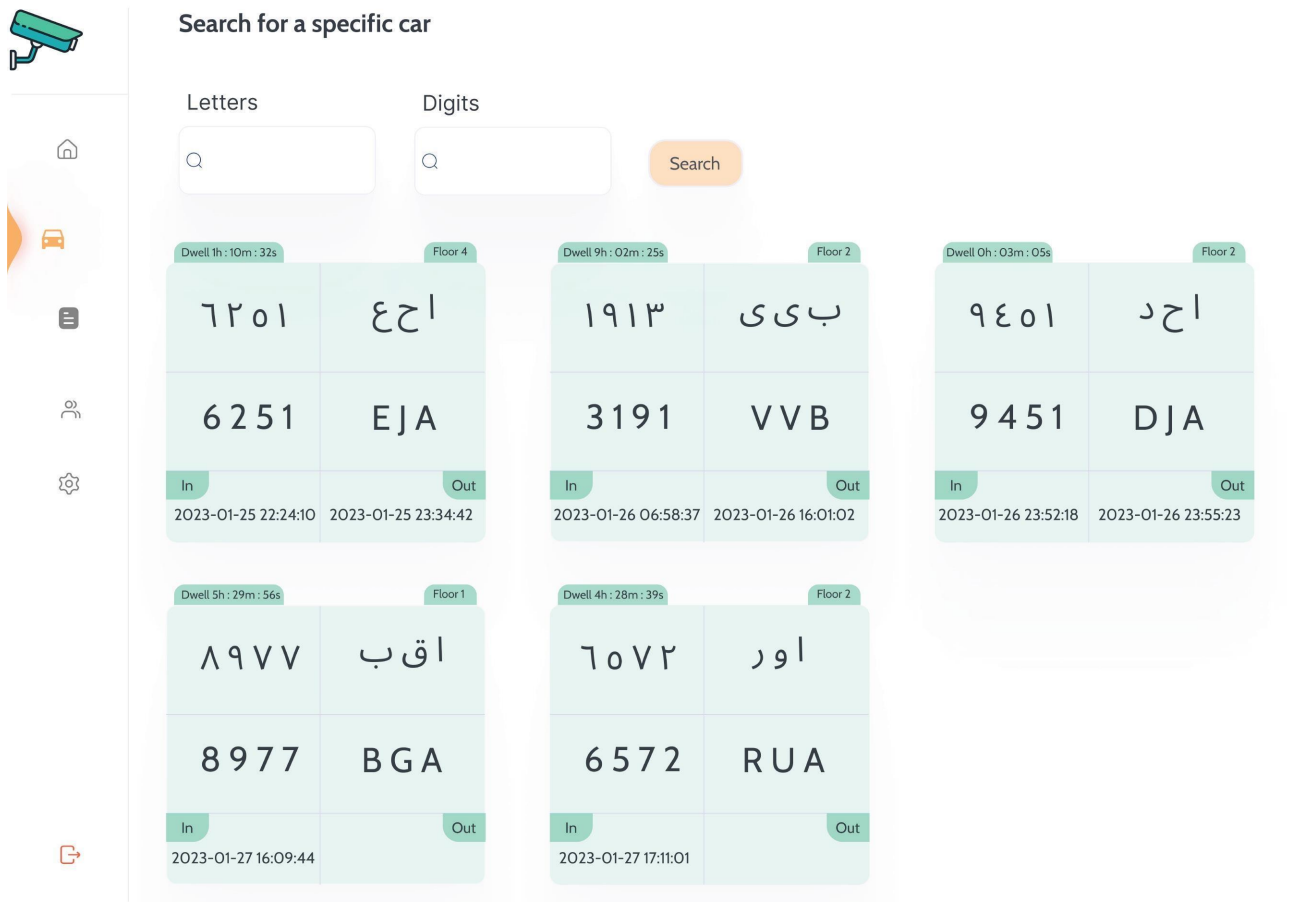
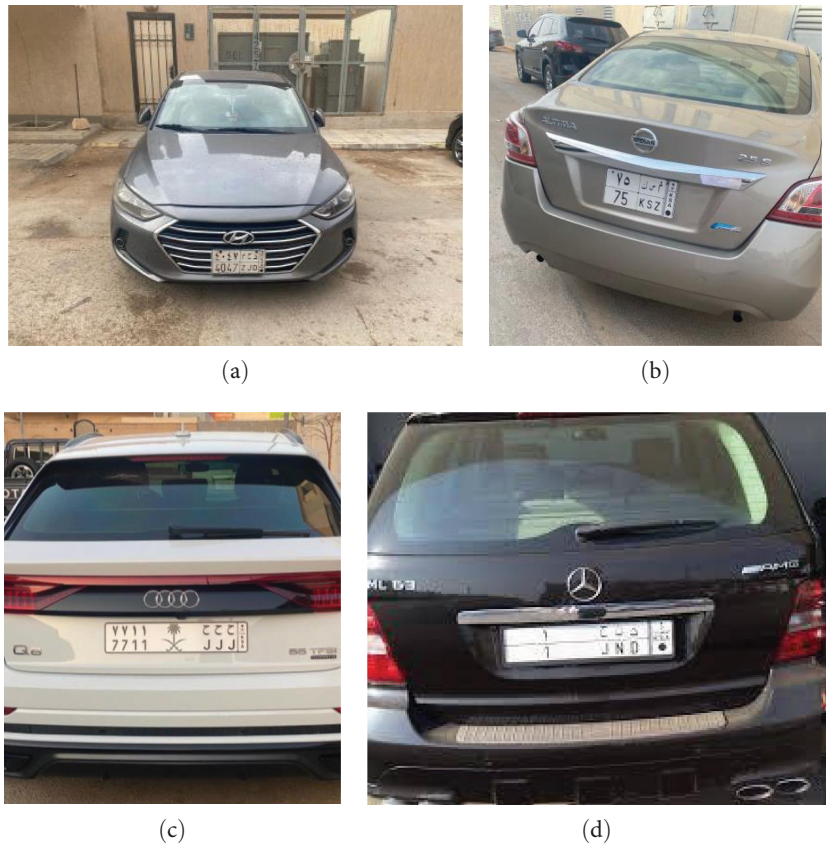


Figure 11: A specific car localization page.

the current number of parked cars, the total number of cars parked during the selected time period, the average parking time per car, and the number of floors and gates in the selected car park (all displayed in blue boxes). The operator can easily compare the car park's capacity with the number of parked cars during the selected time period using a graph at the bottom of the page. The x-axis shows the days, the y-axis shows the total number of parked cars (green columns), and the yellow columns show the capacity of the selected car park or floor. If no filters are set, the default car park's results for the current week are displayed.

(ii) Locate a car: The "Locate a car" page, shown in Figure 11, displays statistics about cars entering and exiting the car park. Unless a specific car is specified in the filter, this page normally displays the last 10 cars that entered and exited. The operator can search for a particular car by entering its license plate letters and digits. Each car has its own profile with information such as its license plate letters and digits in both Arabic and English, floor number, dwell time, entry date, and exit date (if applicable). Car profiles are updated in real-time as cars enter and exit.

3.4. Experiment Preparation and Setup. In our experiments, we used a self-collected dataset of 2,528 images of Saudi car license plates. The images were captured using an iPhone 13 Pro camera at different times of day (morning, afternoon, and evening) and from different positions (front and back license plates), with each image containing only one license plate. The dataset includes a mix of two sizes: 335 mm × 155 mm (1,688 images) and 550 mm × 110 mm (840 images). Figure 12 shows examples of the collected Saudi car license plates in both sizes. We have incorporated real-world distortions into our dataset to make it sufficiently representative of the conditions in a typical car park. Specifically, we have applied a series of transformations to our images to simulate the effects of lighting, weather, and distance variations that a typical surveillance camera in a car park would encounter. These transformations include brightness adjustments, noise addition, and perspective transformations, and also involve resizing the images to 640 × 640 pixels and converting them to grayscale. In addition to these transformations, we have also performed data augmentation on our training images, which include techniques such as HSV adjustment, translation, scaling, horizontal flipping, and mosaic. These techniques further enhance the robustness of our models by providing a diverse set of training samples that cover a wide range of real-world scenarios. By incorporating these real-world distortions, we believe our



(a) (b) (c) (d)

Figure 12: Examples of collected Saudi car license plates: (a and b) size of 335 mm × 155 mm and (c and d) size of 550 mm × 110 mm.

Figure 13: Examples from the dataset used for character detection.

dataset provides a sufficiently realistic representation of real-world scenarios and the challenges that our proposed solution would confront in a practical deployment scenario.

We manually annotated the license plates in the dataset using the LabelImg tool [42] in TXT format to meet the YOLO series standard. For character detection, we manually annotated the regions of interest (bounding boxes on the English letters and Arabic digits) in 1,849 instances from the main dataset. Figure 13 shows some instances from the character detection dataset. For character recognition, we manually annotated the detected characters to create a new balanced dataset containing 27,000 characters with their ground truth labels. Figure 14 shows some instances from the character recognition dataset. Table 2 summarizes the sizes of the main dataset used for license plate detection and the two datasets used for character detection and recognition.

We conducted our experiments on a Windows 10 system with PyTorch1.13 [43] and Tensorflow2.8 [44] frameworks. We used a NVIDIA GeForce RTX 4080 Ti video card with 16 GB of video memory and 64 GB of RAM. Table 3 summarizes the experimental environment configuration.

Figure 14: Examples from the dataset used for character recognition.

Table 2: Number of instances in each dataset.

Dataset	Number of instances
License plate detection	2,528
Character detection	1,849
Character recognition	27,000

Table 3: Experimental environment configuration.

Parameter	Configuration
CPU	Intel Core i9-13980HX (13th Gen)
GPU	NVIDIA GeForce RTX 4080 Ti
System environment	Windows 10
Acceleration environment	CUDA 11.7
Language	Python3.11.3

Table 4: Hyperparameters used for training YOLOv5x, YOLOv7x, and YOLOv8x.

Model	Image size	Batch size	Epochs	Loss	Learning rate	Optimizer	Augmentation
YOLOv5x YOLOv7x YOLOv8x	640 × 640	16	50	0.02	0.01	SGD (0.937 momentum)	hsv (h: 0.015; s: 0.7, v: 0.4), translate: 0.1, scale: 0.5, flip left–right: 0.5, mosaic: 1.0

3.5. Evaluation Metrics. To evaluate the performance of object detection models, several metrics are commonly used, such as precision, recall, average precision, and mean average precision. These metrics are based on the concept of intersection over union (IoU), which measures how much two bounding boxes overlap and it determines whether a prediction is correct or not. The IoU threshold is used to decide whether a prediction is considered a true positive (TP) or a false positive (FP). A prediction is a TP if its IoU with the ground truth box is above the threshold and an FP otherwise. The IoU threshold determines how strictly the model is evaluated in terms of detection accuracy. A higher threshold requires a more precise localization of the object to be a TP, while a lower threshold allows for more leniency in the localization.

Precision (P) is the ratio of TP to all predicted positives (TP + FP), and it reflects how accurate the model is in detecting objects. Recall (R) is the ratio of TP to all ground truth positives (TP + FN), where FN is false negatives, and it reflects how complete the model is in detecting objects. Precision and recall can be computed using the following equations, respectively:

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{1}$$

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{2}$$

AP is the average of precision values at different recall levels for each class, and it reflects how well the model can detect objects across various confidence thresholds. It can be computed using the following equation:

$$\mathrm{AP} = \int_0^1 P(R)\, dR, \tag{3}$$

where $P(R)$ is the precision at a given recall level R.

Mean average precision (mAP) is the mean of AP values for all classes, and it summarizes the overall performance of the model for multiple classes. It can be computed using the following equation:

$$\mathrm{mAP} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{AP}_i, \tag{4}$$

where N represents the number of classes, and AP_i is the average precision for class i.

Table 5: Evaluation results of license plate detection.

Model	mAP@0.5	mAP@0.95
YOLOv8x	0.973	0.844
YOLOv7x	0.920	0.730
YOLOv5x	0.994	0.892

In this study, we used two types of mAP:

(i) mAP@0.5: A mAP computed using an IoU threshold of 0.5.

(ii) mAP@0.5: 0.95 (denoted as mAP@0.95 for simpler presentation): A mAP computed using a range of thresholds from 0.5 to 0.95, increasing by 0.05 each time. This is a stricter metric than mAP@0.5 and is often used to evaluate models that need high detection accuracy, such as license plate detection.

4. Results and Discussion

In this section, we evaluate the performance of our proposed system through various experiments. We first present the training process and evaluation results for each stage of our system: license plate detection, character detection, and character recognition. We then compare our multistage approach with the single-stage approach and analyze their accuracy. We also demonstrate the practicality of our system by implementing it in a real-world scenario and analyzing its time efficiency. Finally, we discuss the limitations and future work of our system.

4.1. Experimental Results

4.1.1. License Plate Detection. For license plate detection, we used our self-collected dataset of 2,528 images of Saudi car license plates. The dataset was divided into training (70%, 1,769 images), validation (20%, 506 images), and testing (10%, 253 images) sets. We trained and finetuned the YOLOv5x, YOLOv7x, and YOLOv8x frameworks with hyperparameters' values given in Table 4. We evaluated these models on the test set, and the accuracy results are presented in Table 5. YOLOv5x outperformed the other two models in detecting license plates with mAP@0.5 of 0.994 and mAP@0.95 of 0.892. Therefore, we chose YOLOv5x for license plate detection in our final approach. Figure 15 shows the training and validation loss versus the epoch of the YOLOv5x for license plate detection.

4.1.2. Character Detection. We annotated the regions of interest (English letters and Arabic digits) in 1,849 license plates detected by YOLOv5x in the previous phase. We

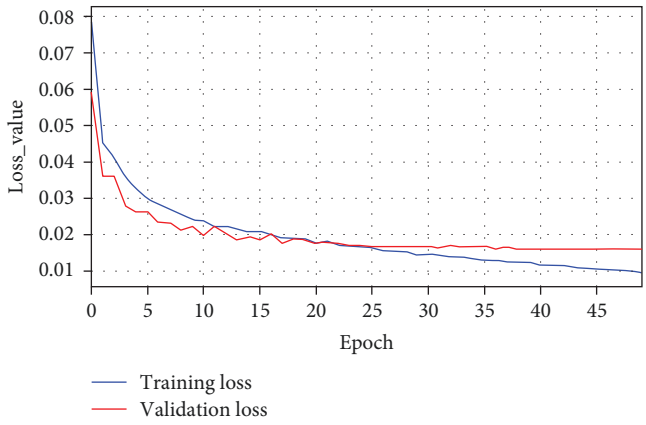


Figure 15: Training and validation loss versus epoch of the YOLOv5x for license plate detection.

Table 6: Evaluation results of character detection.

Model	mAP@0.5	mAP@0.95
YOLOv8x	0.981	0.827
YOLOv7x	0.977	0.711
YOLOv5x	0.978	0.819

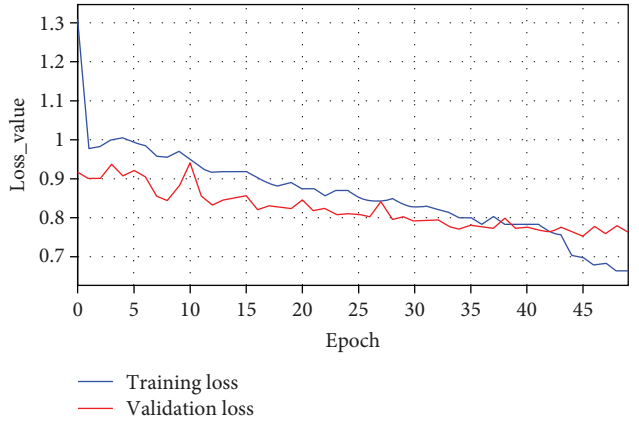


Figure 16: Training and validation loss versus epoch of the YOLOv8x for character detection.

created a dataset of these annotated images and split it into training (70%, 1,295 images), validation (20%, 370 images), and testing (10%, 184 images) sets. We used the same training settings as before to train and finetune the YOLOv5x, YOLOv7x, and YOLOv8x frameworks for character detection. We tested these models on the test set, and Table 6 shows the accuracy results. YOLOv8x achieved the best performance for character detection with mAP@0.5 of 0.981 and mAP@0.95 of 0.827. Figure 16 plots the training and validation loss versus epoch for the YOLOv8x model.

The trained models generate nine bounding boxes, each with its own set of coordinates. To ensure that the segmented elements are characters or digits and to preserve their order on the license plate, we compare the coordinates of the resulting bounding boxes. This allows us to determine which main bounding box the segmented characters and digits belong to. Additionally, we compare the coordinates of each segmented character and digit within a single main bounding box to maintain the correct order of the characters and digits on the license plate.

Table 7: Hyperparameters used to train the proposed CNN model.

Training component	Approach and values
Number of epochs	50 with early stopping
Batch size	16
Learning rate	0.01
Loss function	Categorical Cross-entropy
Weight decay	0.01
Optimizer	Adam

4.1.3. Character Recognition. For character recognition, we manually annotated the detected characters (English letters and Arabic digits) from the previous phase to create a new dataset containing 27,000 characters with their ground truth labels. The dataset was divided into three subsets: training (70%, 18,900 images), validation (20%, 5,400 images), and testing (10%, 2,700 images). Each subset had a balanced distribution of images among the digits and letters.

We trained our CNN framework for character recognition on our dataset with hyperparameters given in Table 7 (including a learning rate of 0.01, which was chosen based on initial experimentation that showed good model generalization with this value). As shown in Figures 17 and 18, our CNN model exhibits high training and validation accuracy and low loss versus epochs. It achieved an average accuracy of 0.970, precision of 0.985, recall of 0.985, and F1-score of 0.982. Table 8 further presents a detailed comparison of the recall, precision, and F1 scores for all classes, and Figure 19 illustrates the performance using a confusion matrix.

4.2. Advantages of the Proposed Multistage Approach over a Single-Stage Approach. This study aims to compare the performance of a multistage approach and a single-stage approach for real-time SPSs. We use different versions of YOLO frameworks (including YOLOv8, which has not been applied to this domain before) for the comparison. We could not find any suitable techniques from the prior art that match our criteria, as most of them are designed for different license plate formats or other applications. Therefore, this work, along with its proposed dataset, could pave the way for more advanced and affordable solutions for SPSs for diverse application scenarios.

We compared the accuracy and efficiency of our proposed multistage approach, which uses YOLOv5x and YOLOv8x for license plate and character detection, respectively, and a separate classification CNN for character recognition, with a single-stage approach that uses YOLOv8x for both detection and recognition tasks. We conducted this experiment using 181 new images and measured the overall performance of the two approaches.

YOLOv5x and YOLOv8x are multiclass detectors that need minimal training for customization to a specific detection

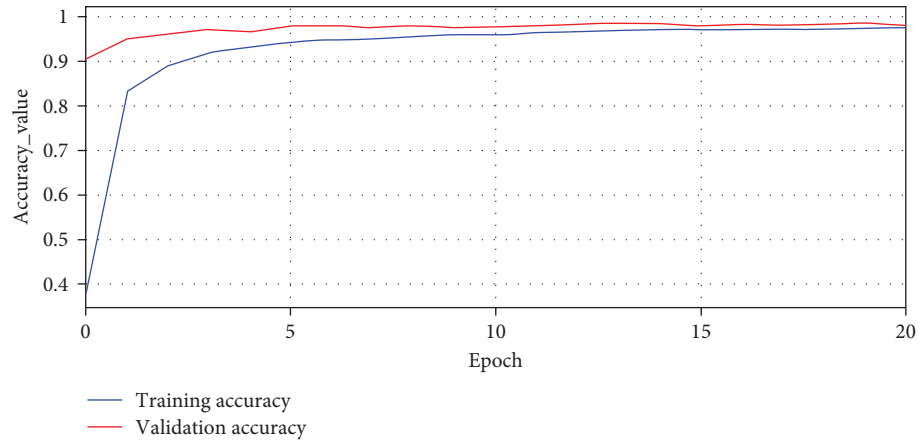


Figure 17: Training and validation accuracy versus epoch of the proposed CNN for character recognition.

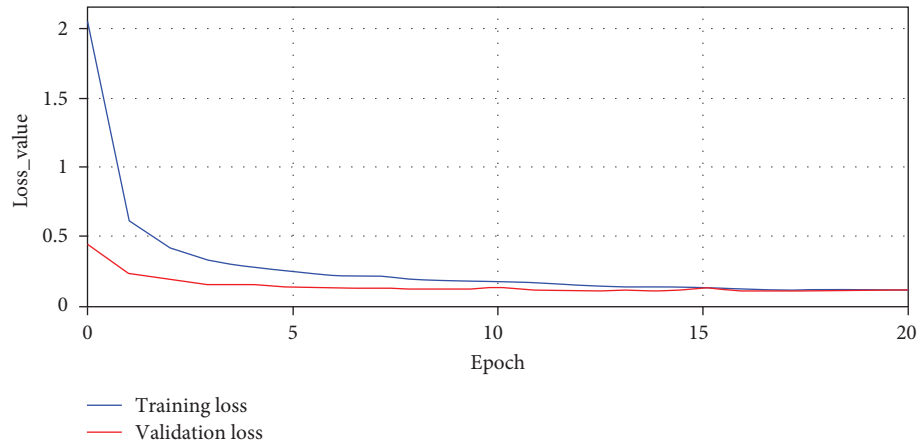


Figure 18: Training and validation loss versus epoch of the proposed CNN for character recognition.

Table 8: Evaluation of character classification of proposed CNN.

Character	Precision	Recall	F1-score
0	0.98	0.99	0.98
1	0.95	0.98	0.96
2	0.96	0.97	0.96
3	0.99	0.98	0.98
4	1	0.98	0.98
5	0.99	0.99	0.99
6	0.97	0.98	0.97
7	0.97	0.97	0.97
8	0.99	0.98	0.98
9	0.99	0.97	0.98
A	1	0.98	0.99
B	1	1	1
D	0.99	0.97	0.98
E	0.98	0.99	0.98
G	1	0.98	0.99
H	0.98	1	0.99
J	0.97	0.96	0.96
K	0.99	0.99	0.99

Table 8: Continued.

Character	Precision	Recall	F1-score
L	0.97	1	0.98
N	0.99	0.99	0.99
R	0.99	1	0.99
S	0.99	0.98	0.98
T	1	1	1
U	0.99	0.99	0.99
V	1	0.99	0.99
X	0.99	0.99	0.99
Z	0.98	1	0.99

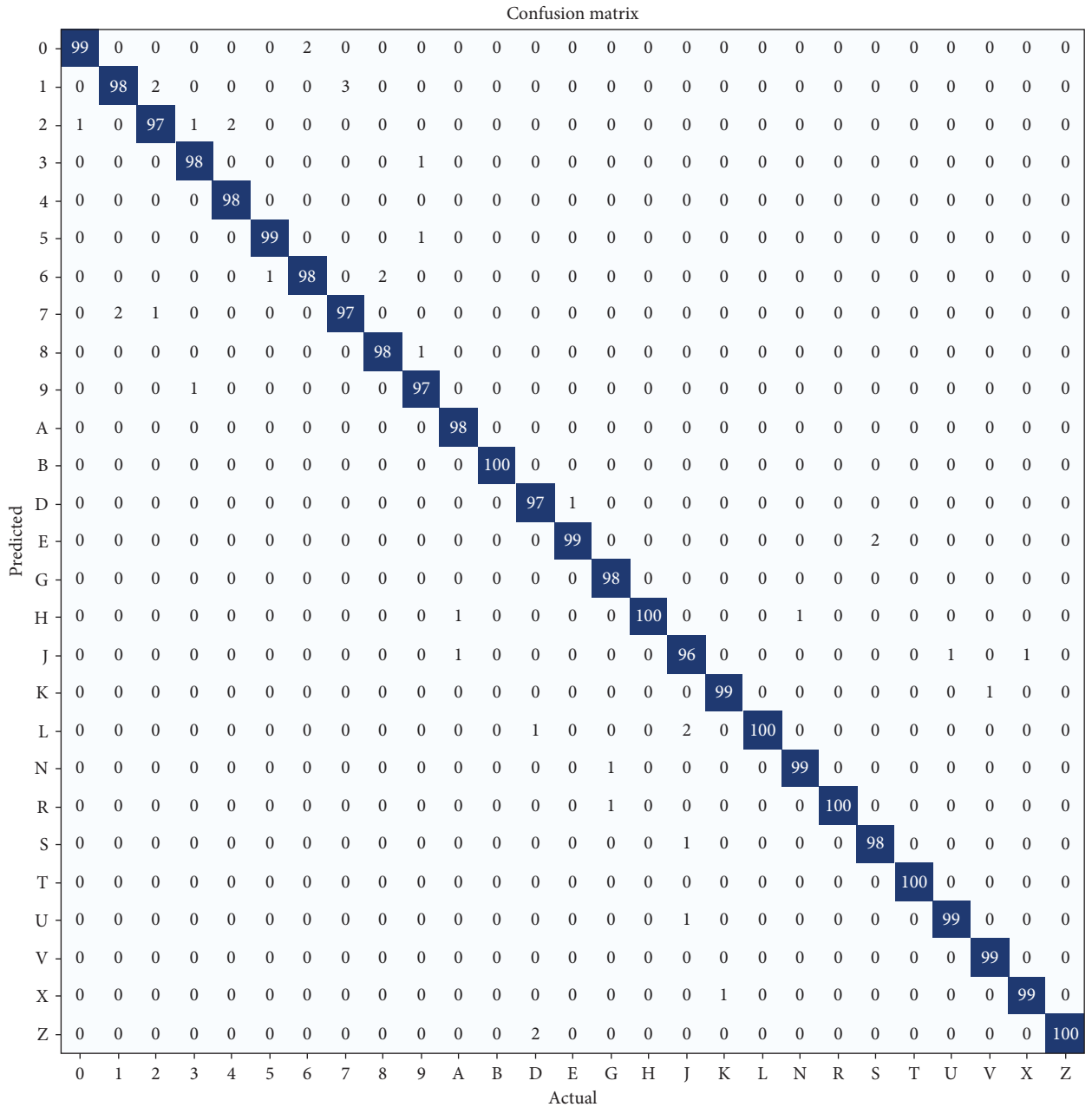


Figure 19: Confusion matrix of proposed CNN for character recognition.

Table 9: Overall accuracy of the proposed multistage approach versus the single-stage approach.

	Single-stage approach	Proposed approach
Number of license plate images	181	181
Correctly detected license plates	179 (98.8%)	179 (98.8%)
Fully correctly recognized license plates	152 (83.9%)	174 (96.1%)

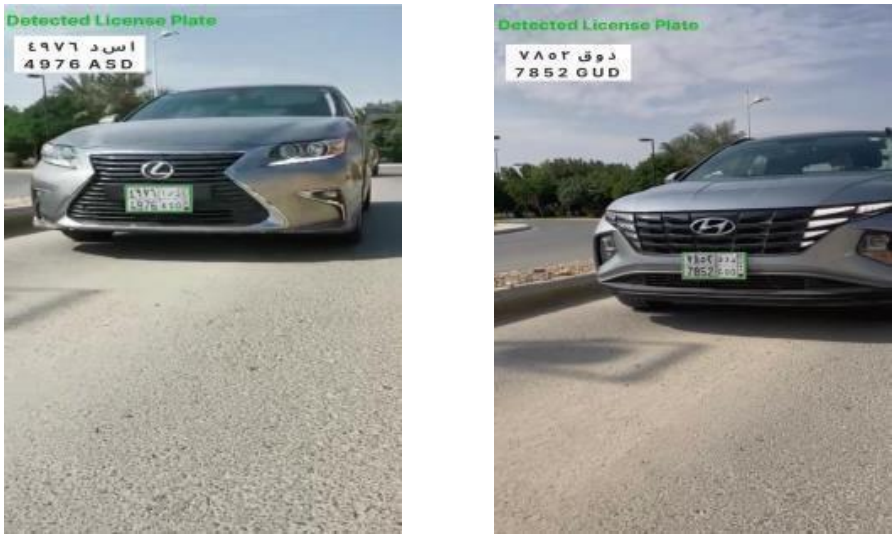


Figure 20: Examples of license plate detection and recognition in a real-world car park using our proposed system.

task, such as license plate detection in our case. By using transfer learning, we trained YOLOv5x and YOLOv8x on our small dataset and achieved a very high detection accuracy of 98.8%. However, while YOLOv5x and YOLOv8x are effective for detection tasks with minimal training, they require more intensive training for recognition tasks on a large dataset. This is because the recognition task involves 27 classes (17 letters and 10 digits) and requires more fine-grained features to distinguish between similar characters, such as O and 0 or B and 8. On the other hand, our proposed approach uses a smaller customized CNN that is specifically designed and trained for the recognition task on our limited dataset. The CNN has fewer parameters and layers than YOLOv8x and uses dropout and regularization to prevent overfitting. The CNN also takes advantage of the character detection results from YOLOv8x, which provide a good localization and segmentation of the characters.

As shown in Table 9, our proposed approach achieved an overall accuracy of 96.1%, which is 12.2% higher than the single-stage approach with 83.9% accuracy. The single-stage approach suffers from low recognition accuracy, especially for small and blurry characters, due to the lack of sufficient training data and the complexity of the recognition task. Therefore, we opted for our proposed two-stage approach instead of using YOLOv8x or YOLOv5x algorithms for both detection and recognition tasks.

4.3. Evaluating the Proposed System in a Real-World Car Park. To evaluate the practical performance of the proposed system, a real-world experiment was conducted in a car park in Saudi Arabia. A camera was mounted at the entrance of the car park and connected to a local PC equipped with a GeForce RTX 2050 GPU. The proposed model, deployed on the local PC, analyzed frames from the camera footage and sent the results to the dashboard via APIs. Figure 20 shows some images from the real-world experiment.

We also measured the time required to process each frame on the local PC and send the results to the dashboard via API. In addition to its high accuracy, our proposed approach demonstrated acceptable time efficiency for use in a parking management system, where cars must slow down at the entrance. Our approach also implemented a redundancy check mechanism to avoid processing the same car multiple times. The mechanism compared the license plate number of the current frame with a buffer of previous frames. If there was a match, the model skipped the current frame. Otherwise, the model sent the current frame to the Cloud Agent and the Client Agent via APIs and updated the buffer. This feature ensured that the model only sent and processed new or updated information. Experimental results showed that the entire process, including license plate recognition, took an average of 264.7 ms/frame.

The potential enhancement of our system's performance through the introduction of real-world distortions to our dataset is noteworthy. Given that our model operates with relatively small image sizes, denoising the images enables it to work effectively with a variety of real-world cameras. Our preprocessing techniques are designed to mitigate these distortions, thereby ensuring robust performance even under challenging conditions. This real-world study not only

demonstrated the effectiveness and efficiency of our proposed system but also highlighted its potential resilience to real-world distortions, further validating its suitability for practical deployment in SPSs.

5. Limitations and Future Work

Our system has some limitations that we acknowledge and plan to address in our future work. These are: (i) Our system relies on the quality and availability of the surveillance cameras within the car park. If the cameras are not well-positioned, calibrated, or maintained, the system may fail to detect or recognize the license plates correctly. Therefore, we need to ensure that the cameras are properly installed and monitored. (ii) Our system may not be able to handle some challenging scenarios, such as occluded, damaged, or fake license plates. These scenarios may require more advanced techniques, such as attention mechanisms, adversarial learning, or anomaly detection, to improve the robustness and security of the system. (iii) Our system is currently designed for Saudi license plates in particular, which have a specific format and character set. To generalize our system to other countries or regions, we need to collect and annotate more data from different license plate types and adapt our models accordingly. In addition to these limitations, we also identify some opportunities for future work. These are: (i) To integrate additional data sources, such as motorcycle license plates and special license plates, to further improve the accuracy and functionality of the system. (ii) To conduct a large-scale deployment of the system in a real-world setting, such as a car park with multiple floors and gates, to evaluate its effectiveness in practice through real experiments. (iii) To extend our comparison to other state-of-the-art algorithms (besides YOLO models) and to test our approach on different license plate types and domains in the future. These directions would help us to enhance the performance and applicability of our methods for smart parking systems.

6. Conclusions

This article proposes a solution for SPSs that addresses the challenges of high installation and maintenance costs and limited functionality in tracking vehicle movement within car parks. The proposed approach leverages existing surveillance cameras and a self-collected dataset of Saudi license plates. The approach trains and fine-tunes state-of-the-art YOLO series for accurate Saudi car license plate detection and recognition, with YOLOv5x achieving mAP@0.5 of 99.4% and mAP@0.95 of 89.2% for license plate detection and YOLOv8x achieving mAP@0.5 of 98.1% and mAP@0.95 of 82.7% for character detection. A new CNN architecture is introduced for improved license plate character recognition, achieving an average accuracy of 97%, average precision of 98.5%, average recall of 98.5%, and average F1-score of 98.2%. Experimental results show that the proposed approach is more effective than the single-stage approach, with overall accuracy of 96.1% compared to overall accuracy of 83.9% for our proposed approach and the single-stage approach, respectively. The proposed approach is also integrated into a web-based dashboard for real-time visualization and statistical analysis, with an acceptable time efficiency of 264.7 ms/frame. This approach has the potential to enhance the efficiency and sustainability of smart cities through its innovative use of existing technology.

Data Availability

The weights and scripts of the proposed models are publicly available through https://github.com/abdulmalik-99/Efficient-Multi-Stage-License-Plate-Detection-and-Recognition. The data presented in this study are available upon request from the corresponding author.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors extend their appreciation to King Saud University for funding this research through researchers supporting project number (RSPD2024R1027), King Saud University, Riyadh, Saudi Arabia.

References

[1] R. Nithya, V. Priya, C. S. Kumar, J. Dheeba, and K. A. Chandraprabha, "Smart parking system: an IoT based computer vision approach for free parking spot detection using faster R-CNN with YOLOv3 method," *Wireless Personal Communications*, vol. 125, no. 4, pp. 3205–3225, 2022.

[2] M. G. D. Ogás, R. Fabregat, and S. Aciar, "Survey of smart parking systems," *Applied Sciences*, vol. 10, no. 3872, 2020.

[3] C. J. Rodier, S. A. Shaheen, and C. Kemmerer, "Smart parking management field test: a Bay Area rapid transit (BART) district parking demonstration," University of California, Berkeley, Final report, 2008.

[4] M. Y. I. Idris, Y. Y. Leng, E. M. Tamil, N. M. Noor, and Z. Razak, "Car park system: a review of smart parking system and its technology," *Information Technology Journal*, vol. 8, no. 2, pp. 101–113, 2009.

[5] A. Fahim, M. Hasan, and M. A. Chowdhury, "Smart parking systems: comprehensive review based on various aspects," *Heliyon*, vol. 7, no. 5, Article ID e07050, 2021.

[6] C.-F. Chien, H.-T. Chen, and C.-Y. Lin, "A low-cost on-street parking management system based on bluetooth beacons," *Sensors*, vol. 20, no. 16, Article ID 4559, 2020.

[7] B. Budihala, T. Ivaşcu, and S. tefaniga, "Motorage—computer vision-based self-sufficient smart parking system," in *Proceedings of the 2022 24th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 250–257, IEEE, Linz, Austria, 2022.

[8] M. Dixit, C. Srimathi, R. Doss, S. Loke, and M. Saleemdurai, "Smart parking with computer vision and iot technology," in *Proceedings of the 2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, pp. 170–174, IEEE, Italy, 2020.

[9] T. Sirithinaphong and K. Chamnongthai, "The recognition of car license plate for automatic parking system," *Proceedings of the ISSPA '99. Proceedings of the Fifth International*

Symposium on Signal Processing and Its Applications, vol. 1, pp. 455–457, 1999.

[10] Joshua, J. Hendryli, and D. E. Herwindiati, "Automatic license plate recognition for parking system using convolutional neural networks," in *Proceedings of the 2020 International Conference on Information Management and Technology (ICIMTech)*, pp. 71–74, IEEE, Indonesia, 2020.

[11] N. Darapaneni, K. Mogeraya, S. Mandal et al., "Computer vision based license plate detection for automated vehicle parking management system," in *Proceedings of the 2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, pp. 0800–0805, IEEE, New York, 2020.

[12] N. Thakur, S. M. N. Islam, Z. Neyaz, D. Sadhwani, and R. Jain, "Smart parking system using YOLOv3 deep learning model," in *Applications of Artificial Intelligence, Big Data and Internet of Things in Sustainable Development*, S. Goundar, A. Purwar, and A. Singh, Eds., CRC Press, chapter 4, 2022.

[13] D. Neupane, A. Bhattarai, S. Aryal et al., "Shine: a deep learning-based accessible parking management system," *Expert Systems with Applications*, vol. 238, Article ID 122205, 2023.

[14] M. M. Rashid, A. Musa, M. A. Rahman, N. Farahana, and A. Farhana, "Automatic parking management system and parking fee collection based on number plate recognition," *International Journal of Machine Learning and Computing*, vol. 2, no. 2, pp. 93–97, 2012.

[15] E. J. Sen, K. D. M. Dixon, A. Anto et al., "Advanced license plate recognition system for car parking," in *Proceedings of the 2014 International Conference on Embedded Systems (ICES)*, pp. 162–165, IEEE, India, 2014.

[16] Q. T. Van, H. N. Van, L. D. V. Hoang et al., "Intelligent parking system using automated license plate recognition and face verification," *Proceedings of the Proceedings of International Conference on Computing and Communication Networks; Springer Nature Singapore*, pp. 219–227, 2022.

[17] N. Thai-Nghe and N. Chi-Ngon, "An approach for building an intelligent parking support system," in *Proceedings of the Proceedings of the 5th Symposium on Information and Communication Technology; Association for Computing Machinery*, pp. 192–201, Association for Computing Machinery, New York, NY, USA, 2014.

[18] L. Shkurti, A. Aliu, and F. Kabashi, "ParkingKS: parking management system using open automatic license plate recognition," in *Proceedings of the 2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pp. 1–5, IEEE, Cape Town, South Africa, 2021.

[19] H. T. Thai, T. L. Nguyen-Tran, and K. H. Le, "Toward a predictive smart parking system in IoT-enabled cities," in *Proceedings of the 2022 9th NAFOSTED Conference on Information and Computer Science (NICS)*, pp. 1–6, 2022.

[20] N. Dalarmelina, M. A. Teixeira, and R. I. Meneguette, "A real-time automatic plate recognition system based on optical character recognition and wireless sensor networks for ITS," *Sensors*, vol. 20, no. 1, Article ID 55, 2020.

[21] Z. Xiang and J. Pan, "Design of intelligent parking management system based on ARM and wireless sensor network," *Mobile Information Systems*, vol. 2022, Article ID 2965638, 13 pages, 2022.

[22] F. Anuar and N. Lingas, "Smart campus initiative: car entrance, exit and parking management prototype development," *AIP Conference Proceedings*, vol. 2643, Article ID 040029, 2023.

[23] T. Saidani and Y. E. Touati, "Vehicle plate recognition system based on deep learning algorithms," *Multimedia Tools and Applications*, vol. 80, no. 30, pp. 36237–36248, 2021.

[24] I. R. Khan, S. T. A. Ali, A. Siddiq et al., "Automatic license plate recognition in real-world traffic videos captured in unconstrained environment by a mobile camera," *Electronics*, vol. 11, no. 9, Article ID 1408, 2022.

[25] A. Ammar, A. Koubaa, W. Boulila, B. Benjdira, and Y. Alhabashi, "A multi-stage deep-learning-based vehicle and license plate recognition system with real-time edge inference," *Sensors*, vol. 23, no. 4, Article ID 2120, 2023.

[26] I. R. Khan, S. T. A. Ali, A. Siddiq, and S. O. Shim, "Multi-string missing characters restoration for automatic license plate recognition system," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 95, 2023.

[27] M. Driss, I. Almomani, R. Al-Suhaimi, and H. Al-Harbi, "Automatic Saudi Arabian license plate detection and recognition using deep convolutional neural networks," in *Proceedings of the Advances on Intelligent Informatics and Computing*, pp. 3–15, Springer International Publishing, Cham, 2022.

[28] K. W. Maglad, "A vehicle license plate detection and recognition system," *Journal of Computer Science*, vol. 8, no. 3, pp. 310–315, 2012.

[29] H. M. Alyahya, M. K. Alharthi, A. M. Alattas, and V. Thayananthan, "Saudi license plate recognition system using artificial neural network classifier," in *Proceedings of the 2017 International Conference on Computer and Applications (ICCA)*, pp. 220–226, IEEE, Doha, Qatar, 2017.

[30] K. Suwais, R. Al-Otaibi, and A. Alshahrani, "Saudi license plate recognition algorithm based on support vector machine," *Journal of Electronic Science and Technology*, vol. 11, pp. 424–428, 2013.

[31] F. Kurniawan and M. S. Khalil, "Performance comparison between SVM-based and RBF-based for detection of Saudi license plate," in *Proceedings of the 2012 8th International Conference on Information Science and Digital Content Technology (ICIDT2012)*, pp. 537–541, IEEE, Jeju, Korea (South), 2012.

[32] L. Alzubaidi, G. Latif, and J. Alghazo, "Affordable and portable realtime Saudi license plate recognition using SoC," *Proceedings of the 2019 2nd International Conference on New Trends in Computing Sciences (ICTCS)*, pp. 1–5, 2019.

[33] T.-Y. Lin, M. Maire, S. Belongie et al., "Microsoft COCO: common objects in context," *European Conference on Computer Vision*, vol. 8693, pp. 740–755, 2014.

[34] G. Jocher, "Ultralytics/Yolov5:v3.1—bug fixes and performance improvements," 2020, Zenodo, https://zenodo.org/record/4154370.

[35] C. Wang, A. Bochkovskiy, and H. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1571–1580, IEEE, New Orleans, LA, USA, 2022.

[36] ultralytics/ultralytics, 2023, https://github.com/ultralytics/ultralytics.

[37] C. Y. Wang, H. Y. Mark 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 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1571–1580, IEEE, Seattle, WA, USA, 2020.

[38] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, "PANet: few-shot image semantic segmentation with prototype alignment," in *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9196–9205, IEEE, Seoul, Republic of Korea, 2019.

[39] C. Wang, H. M. Liao, I. Yeh, and E. M. Corporation, "Designing network design strategies through gradient path analysis," arXiv 2014, arXiv: 2211.04800

[40] X. Ding, X. Zhang, J. Han, G. Ding, and J. Sun, "RepVGG: making VGG-style convnets great again," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13728–13737, Long Beach, CA, USA, 2019.

[41] M. Sportelli, O. E. Apolo-Apolo, M. Fontanelli et al., "Evaluation of YOLO object detectors for weed detection in different turfgrass scenarios," *Applied Sciences*, vol. 13, no. 14, Article ID 8502, 2023.

[42] Tzutalin, "LabelImg," 2015, https://github.com/tzutalin/labelImg.

[43] A. Paszke, S. Gross, F. Massa et al., "PyTorch: an imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32; Curran Associates, Inc*, pp. 8024–8035, 2019.

[44] TensorFlow, [Online; accessed 2023-08-05], https://www.tensorflow.org.