Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 10, No. 4, November, Pp. 511-524

Implementation of convolutional recurrent neural network for vehicle number plate identification in a Raspberry Pi based parking system

Rivaul Muzammil[1], Maulisa Oktiana[*1], Roslidar Roslidar[1]
Department of Electrical and Computer Engineering, Universitas Syiah Kuala, Banda Aceh, Indonesia[1]

Article Info
Keywords:
Parking system, License Plate Numbers, You Only Look Once v8, Convolutional Recurrent Neural Network, Raspberry Pi

Article history:
Received: April 29, 2025
Accepted: August 20, 2025
Published: November 01, 2025

Cite:
R. Muzammil, M. Oktiana, and Roslidar, "Implementation of Convolutional Recurrent Neural Network for Vehicle Number Plate Identification in a Raspberry Pi Based Parking System", *KINETIK*, vol. 10, no. 4, Nov. 2025.
https://doi.org/10.22219/kinetik.v10i4.2320

*Corresponding author.
Maulisa Oktiana
E-mail address:
maulisaoktiana@usk.ac.id

Abstract
The rapid growth of vehicles in Indonesia has created significant challenges in managing parking facilities. To address this issue, this study proposes an intelligent parking system based on automatic license plate character recognition. The system employs YOLOv8 (You Only Look Once) for license plate region detection and CRNN (Convolutional Recurrent Neural Network) for alphanumeric character recognition. Its architecture integrates a Raspberry Pi, camera module, and servo motor to enable automated license plate detection and recognition during vehicle entry and exit. YOLOv8 generates bounding boxes to isolate license plate regions, which are then processed as input for CRNN. The CRNN extracts visual features through convolutional layers and captures sequential relationships among characters using recurrent layers. The entire pipeline is deployed on Raspberry Pi with TensorFlow Lite to ensure efficient computation in resource-constrained environments. Experimental results demonstrate that YOLOv8 achieved a detection accuracy of 94.69%, with a precision of 98.32%, recall of 96.25%, and F1-score of 97.27%, while CRNN reached a character recognition accuracy of 93.8% across 30 license plates. Although some recognition errors occurred, such as misclassifying 'G' as 'C', 'W' as 'H', and 'Q' as 'O', the proposed system proved effective and feasible for embedded smart parking applications.

1. Introduction

Vehicles have become an essential necessity in modern society, particularly in line with rapid population growth, leading to a continuous increase in their numbers. According to data from the Central Bureau of Statistics, there are 148,261,817 registered vehicles in Indonesia [1]. Each vehicle, whether two-wheeled or four-wheeled, is required to possess a license plate as an official form of identification, consisting of a unique combination of letters and numbers.

With the growing number of vehicles on the road, issues related to parking management have become increasingly prominent, further aggravated by the inadequacy of conventional parking systems [2]. Manual license plate recording in parking areas is associated with several drawbacks, including human error, prolonged waiting times, inconvenience for drivers, and the potential loss of physical parking tickets. To address these challenges, the implementation of intelligent parking systems has been proposed as an effective solution [3][4], particularly given the high volume of vehicles requiring accurate verification.

In response to the need for improved accuracy and operational efficiency in license plate recognition and parking management, numerous approaches have been explored by researchers. One such method involves plate position comparison based on maximum value detection, which has demonstrated an accuracy rate of 81% in counting vehicles in indoor parking facilities. This technique incorporates several image processing methods, including Retinex enhancement, edge histogram analysis, integral intensity projection, smoothing filters, first-order derivatives, and peak detection, to accurately identify the license plate region [5].

A study conducted in Sudan employed a Convolutional Neural Network (CNN) and achieved an accuracy rate of 90% over 100 test iterations. The study adopted the Region-based Convolutional Neural Network (R-CNN) framework for license plate detection and localization, with the Faster R-CNN variant demonstrating superior accuracy compared to the baseline approach. While the use of CNN significantly enhanced system reliability, the R-CNN architecture imposed high computational demands, thereby limiting its feasibility for deployment on Internet of Things (IoT) devices [4].

Another line of research utilized the Character Region Awareness for Text (CRAFT) method for text detection and the Convolutional Recurrent Neural Network (CRNN) for character recognition, resulting in an overall accuracy of 85.73%. However, real-time implementation revealed challenges in accurately detecting license plate regions. This limitation was attributed to the four-angle contour detection method employed, which is prone to error due to the presence of similar rectangular contours on other parts of the vehicle. Misidentification of these contours can lead to detection failures [6].

Additional studies explored the integration of cameras and infrared sensors [7], as well as IoT-based systems incorporating cloud-integrated QR codes to streamline university parking operations using automated gates and QR-based authentication [8]. Although these approaches achieved commendable levels of accuracy, they remain constrained by high computational complexity, substantial processing power requirements, and limited validation in real-world environments.

This study developed a vehicle license plate recognition system utilizing YOLOv8 and Convolutional Recurrent Neural Network (CRNN) as an automated identity verification mechanism for parking lot entry. The system adopts the You Only Look Once (YOLOv8) model for detecting the license plate region, while the CRNN model is employed to recognize individual characters on the license plate. The hardware implementation is based on a Raspberry Pi microcontroller, which receives image input from a connected camera and controls a servo motor to operate a parking barrier. The system functions in both entry and exit scenarios, where license plates are detected and recognized upon vehicle arrival and departure. Upon exit, the system also provides parking duration information based on the recorded entry and exit timestamps. This research aims to demonstrate that the integration of YOLOv8 and CRNN can achieve optimal recognition accuracy and efficient computation, making it suitable for deployment in resource-constrained environments such as embedded or IoT-based parking systems.

This study presents three main contributions. First, it implements an end-to-end intelligent parking system that integrates real-time license plate detection and character recognition by leveraging deep learning techniques, specifically YOLOv8 and CRNN, on embedded hardware (Raspberry Pi). Second, it introduces a lightweight and low-cost prototype designed to suit smart parking applications in environments with limited computational resources, thereby promoting broader accessibility and scalability. Third, the system offers an automatic parking duration calculation feature, which enhances its functionality and practicality in managing vehicle flow and maintaining accurate parking time records.

The objective of this study is to develop and evaluate an intelligent license plate recognition system for smart parking applications by integrating YOLOv8 for plate detection and CRNN for character recognition on embedded hardware. The system is designed to achieve high recognition accuracy with efficient computation, making it suitable for deployment in resource-constrained environments such as IoT-based or embedded parking systems.

2. Literature Review

2.1 License Plate

A license plate is an official identification mark for a motor vehicle, generally consisting of a combination of letters and numbers. It is a metal or plastic plate attached to the vehicles and function as a valid identification code. Many researchers have investigated the field of image processing and text recognition on license plates, including the best techniques used to extract clean text and improve system accuracy. A study identifies Egyptian car license plates. The image processing stages to achieve high accuracy license plate identification are pre-processing, segmentation, and character recognition. Canny edge detection method with various thresholds, contour detection, and masking techniques are used to find the edges of the car and license plate. A prototype was implemented using an ESP32 camera and a Raspberry Pi to test the performance of the system [3].

Another study mainly focuses on Nepal's vehicle license plate recognition system, in which the vehicle license plate image is captured using a digital camera and then processed to obtain the license plate information [9]. This study used several methods, namely morphological operation, edge detection, smoothing, filtering, plate localization techniques, and character segmentation. The system was tested with 90 patterns under several conditions. This includes license plate recognition experiments using phase correlation and normalized cross-correlation methods. From the study and analysis of the tests after being applied to a number of database images, the normalized cross-correlation method was found to be more accurate for recognizing license plates than the phase correlation method, and achieving a recognition accuracy of 67.98%, compared to 63.46%.

A separate study investigated Indian license plate detection, using You Only Look Once version 5 (YOLOv5) for license plate detection and Google Tesseract for character recognition. Most cars have the same single license plate while motorcycles, buses, rickshaws, etc., usually have multiple license plates. This makes LP detection almost impossible. To overcome this difficulty, a database with different types of images of all license plate patterns that can be found in India was proposed [10].

Furthermore, another paper proposed an algorithm that is optimized to work with vehicle license plates in Ghana. This algorithm, written in C++ with the OpenCV library, uses edge detection and feature detection techniques combined with mathematical morphology to locate license plates. The Tesseract OCR engine is then used to identify the detected characters on the plate [11].

Another paper proposes a methodology for Malaysian license plate segmentation using group thresholding segmentation. Threshold-based image segmentation is chosen because of its ability to separate foreground and background. In this study, threshold techniques such as Sauvola and Niblack have been selected to be compared. The final results show that Sauvola occupies the highest accuracy in license plate recognition [12].

Additionally, in another study, the license plate detection method implemented in this system uses the You Only Look Once (YOLO) approach as a detector of each segment of the license plate code. Furthermore, the Convolutional Recurrent Neural Network (CRNN) technique is applied with the aim of increasing the accuracy of recognizing plate characters.

2.2 YOLO (You Look Only Once)

YOLO (You Only Look Once) is a widely implemented real-time object detection algorithm. According to research comparing YOLO with its previous versions, YOLOv8 shows an increase in mean Average Precision (mAP) which is higher than the previous versions [13]. One study presents an innovative approach to improve the accuracy of the YOLOv8 model in object detection, with the main focus on overcoming the limitations when detecting objects in different types of images, especially for small objects. The strategy incorporated a Context Attention Block (CAB) to effectively locate and identify small objects in images. The improved YOLOv8 model, referred to as YOLOv8-CAB, strongly emphasizes the performance in detecting smaller objects by utilizing CAB blocks to exploit multi-scale feature maps and iterative feedback, thus optimizing the object detection mechanism [14].

Another study designed a C2f-D-Mixer (C2f-DM) module as a replacement for the original C2f module, integrating both local and global information to significantly improve the ability to detect small object features [15]. The advantage of this algorithm is that it not only has higher precision for small-sized object but also ensures that the detection accuracy for each size is not lower than that of existing algorithms [16].

A separate study presented an advanced Automatic Number Plate recognition (ANPR) system, specifically designed for Qatar's diverse license plate format and challenging environmental conditions. In this work, the YOLOv8 deep learning model, specifically the YOLOv8s variant [17], was employed to detect the locations of vehicle license plates in car and motorcycle images.

2.3 CRNN (Convolutional Recurrent Neural Network)

CRNN (Convolutional Recurrent Neural Network) is a hybrid architecture that combines two different artificial neural network models, namely CNN (Convolutional Neural Network) and RNN (Recurrent Neural Network). In the CRNN architecture, CNN functions as a feature extractor to identify important features from the input data, while RNN serves as a sequential processor responsible for modeling the contextual dependencies among the features extracted by the CNN [18]. Deep Neural Network models, including CNNs and RNNs, provide promising results on text classification tasks. RNN-based architectures, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), can process long sequences [19]. A combined architecture of CNN and RNN, which utilizes coarse local features generated by CNN and long-range dependencies learned through RNN for sentiment analysis of short texts. Experimental results show a clear improvement over existing methods on three benchmarks, namely MR, SST1, and SST2, with accuracies of 82.28%, 51.50%, and 89.95%, respectively [20].

Another study improved the CRNN model for text recognition, addressing challenges related to low accuracy and incomplete information acquisition. To solve these problems the study added label smoothing to ensure the model's generalization ability. The study also introduced a loss smoothing function of speech recognition into the field of text recognition domain and incorporated a language model to improve the information acquisition process, which ultimately achieves the purpose of improving the accuracy of text recognition [21].

A separate paper proposed a CRNN-based text recognition framework that combines real-time scene text detection with differentiated binarization (DBNet) for text detection and segmentation, text-direction classifier, and the Retinex algorithm for image enhancement. Experimental results show that the proposed model provides better results, successfully recognizing text in complex and multi-oriented text scenes. Moreover, it outperforms the original CRNN model with higher accuracy across various application scenarios [22].

CRNN has some distinctive advantages compared to conventional neural network models. It can be trained directly using sequence-level labels (e.g., words) without requiring detailed annotations (e.g., characters). It has the same properties as DCNN in learning informative representations directly from image data, eliminating the need for handcrafted features or preprocessing steps such as binarization, segmentation, component localization, and others. CRNN is not limited by the length of object sequences and only requires proper normalization during the training and testing phases. Additionally, it achieves better or highly competitive performance in word recognition compared to previous models [23].

The Convolutional Recurrent Neural Network (CRNN) model in this study was built to perform character recognition on detected license plates. CRNN is an artificial neural network architecture that combines the ability of Convolutional Neural Network (CNN) in extracting spatial features from images with Recurrent Neural Network (RNN) that can learn sequential context from text.

2.4 Raspberry Pi

Raspberry Pi is a Single Board Computer (SBC) that has dimensions resembling a credit card. This device is equipped with all the functions of a complete computer, using an ARM System-on-Chip (SoC) integrated on a Printed Circuit Board (PCB). Raspberry Pi is highly useful in prototyping due to its compact size, affordability, and flexibility. In the context of prototyping, it serves as a platform for developing, testing, and validating concepts and ideas.

One study developed a real-time parking monitoring system that can effectively manage parking spaces. To achieve this goal, a Raspberry Pi-based system was designed and implemented by incorporating IR sensors, servo motors, and an LCD display. The system utilizes the MQTT protocol to enable real-time data transmission and integration with mobile applications [24]. The results show that the proposed system can accurately monitor the parking space availability, control access to the parking area, and provide users with a convenient and efficient parking experience. Additionally, empty slot detection is carried out using IR sensors, and payment calculation is performed based on the parking duration [25].

Another real-time parking management system was developed by integrating GPS coordinates through a smartphone application. The system consists of a Raspberry Pi 4 B+ (RPi), a Pi camera module, a GPS sensor, and an ultrasonic sensor. The RPi 4 B+ collects and processes input data from the sensors/camera, and the data is uploaded via Wi-Fi to the Blynk IoT server. Ultrasonic and LED sensors are used to detect parking space availability, and the Pi camera is used to ensure data accuracy [26].

Another study presented an automated vehicle license plate recognition system using Raspberry Pi. A camera captures images of vehicle license plate number, which are then processed by the Raspberry Pi processor for authentication. By applying Open Computer Vision (Open CV) and Optical Character Recognition (OCR), this system can extract numbers from captured license plate images and fully automate license plate recognition [27].

In addition, another study uses a license plate detection system in parking lots using entry and exit cameras to detect vehicle license plates. This system is also equipped with a servo motor that controls the parking gate. When the camera detects a vehicle license plate, the servo motor automatically opens the parking barrier. This system is implemented on a Raspberry Pi device.

2.5 Model to Prototype Integration

The YOLOv8 model processes the image through convolution layers and makes predictions in the form of bounding boxes that mark the areas containing vehicle license plates in the image. The coordinates of the bounding box generated by the YOLOv8 model are then used to crop the relevant areas of the original input image. This cropping process results in a new image that contains only the license plate, without any other objects or background. After the license plate image is successfully extracted, the next step is to perform pre-processing on the image to ensure that the input data matches the format expected by the CRNN model. The prepared license plate image is then processed into the CRNN model to perform character recognition.

The CRNN model is a hybrid architecture that combines the strengths of CNN (Convolutional Neural Network) and RNN (Recurrent Neural Network) [28]. The convolution layer in CRNN extracts visual features from the license plate image that represent the characters, while the recurrent layer models the sequential relationship between the characters. The entire process of vehicle license plate detection and recognition is implemented on a Raspberry Pi device. This process is implemented on Raspberry Pi using TensorFlow Lite, which allows YOLOv8 and CRNN to run efficiently. This library converts trained deep learning models into a more efficient format and optimized for Raspberry Pi devices.

3. Method

3.1 Dataset

This research focuses on objects in the form of vehicle license plates and utilizes two types of datasets. The first dataset is used to train the model to detect the license plate area, while the second dataset is used to train the model to recognize the characters on the license plate. The dataset consists of images of the front of vehicles containing visible license plate. For license plate detection, the *License Plate Recognition Image Dataset* [29] is used, while for training the character recognition model, the *Indonesian Plate Number* dataset [30][31] is employed.

3.2 Building Model YOLOv8 (You Only Look Once)

As shown in Figure 1, the process of vehicle license plate detection using the YOLOv8 Object Detector begins by taking a video as input. Each frame in the video is processed using the Image Frame model, which extracts and prepares the frames for the next stage. The frames are then passed to the YOLOv8 Object Detector, which has been trained to detect vehicle license plates. This model analyzes each frame and generates a bounding box that surrounds the detected license plate. The bounding box contains information about the location and dimensions of the license plate within the frame. The output of this process is then forwarded to the next stage, which is character recognition using the CRNN model.

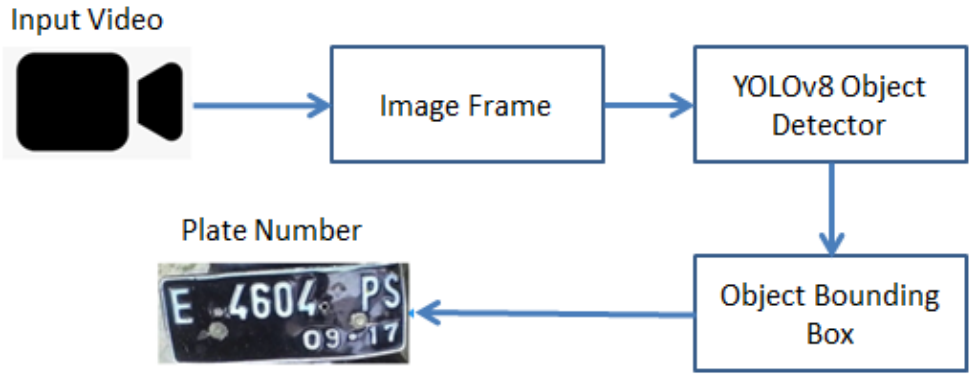


Figure 1. Number Plate Detection Flow Using YOLOv8

3.3 Design and Implementation of the CRNN Model

In developing the CRNN model, the first step is to preprocess the dataset by equalizing the size of the license plate images and performing data augmentation, such as rotation, shifting, and contrast adjustment. This step aims to increase the variety of training data while strengthening the generalization ability of the model. As shown in Figure 2, the CRNN (Convolutional Recurrent Neural Network) architecture for extracting text from vehicle license plates begins with the license plate image as input. This image is processed through several convolutional layers to extract visual features and generate a feature map that captures both spatial structure and pattern information. The feature map is then converted into a sequential feature representation.

Next, this feature sequence is processed by a recurrent layer, such as Bidirectional Long Short-Term Memory (Bi-LSTM) or other recurrent mechanisms, to capture contextual relationships within the feature sequence. The output of the recurrent layer produces a rich sequence representation, which is then used by the transcription layer, such as the fully connected layer, to predict the sequence of characters on the license plate. Finally, the CRNN model outputs the character sequence, representing the text extracted from the license plate image.

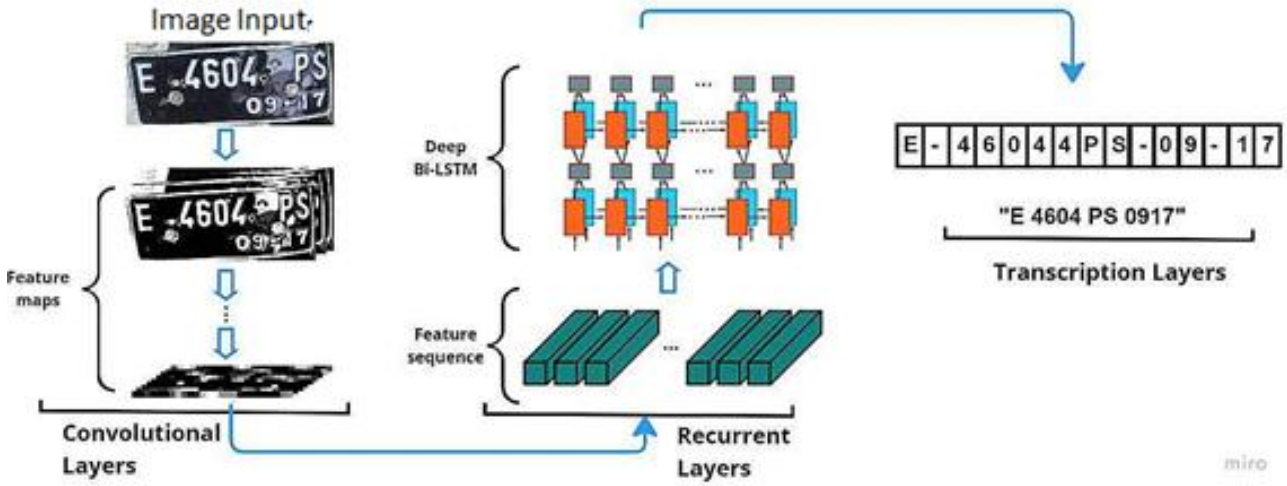


Figure 2. CRNN Architecture

3.4 Analysis of YOLOv8 and CRNN models

This analysis aims to evaluate the performance of the model and identify areas that still need improvement. For license plate detection models using YOLOv8, analysis can be carried out by calculating metrics such as precision, recall, and F1-score. These metrics serve as the basis for optimizing the training process and adjusting model parameters to improve detection performance.

For the license plate character recognition model using CRNN, analysis is conducted by calculating the character recognition accuracy on the testing dataset. This accuracy represents the proportion of correctly recognized characters compared to the total number of characters that should be recognized. In addition to evaluating overall accuracy, character-level analysis is also conducted to identify which characters are most difficult for the model to recognize.

3.5 Building Prototype with Raspberry Pi

At the stage of building the prototype, the camera is used as the source of visual input, the Raspberry Pi acts as the processing unit, and the servo motor operates the gate barrier. Both models are integrated into the prototype, as shown in Figure 3. Image or video input from the camera is first processed by the YOLOv8 model to detect vehicle license plate areas. The area containing the license plate is then extracted and passed to the CRNN model to recognize the characters, producing an output in the form of a character string representing the vehicle's license plate. With the appropriate hardware implementation, this system can perform license plate detection and recognition properly and correctly.

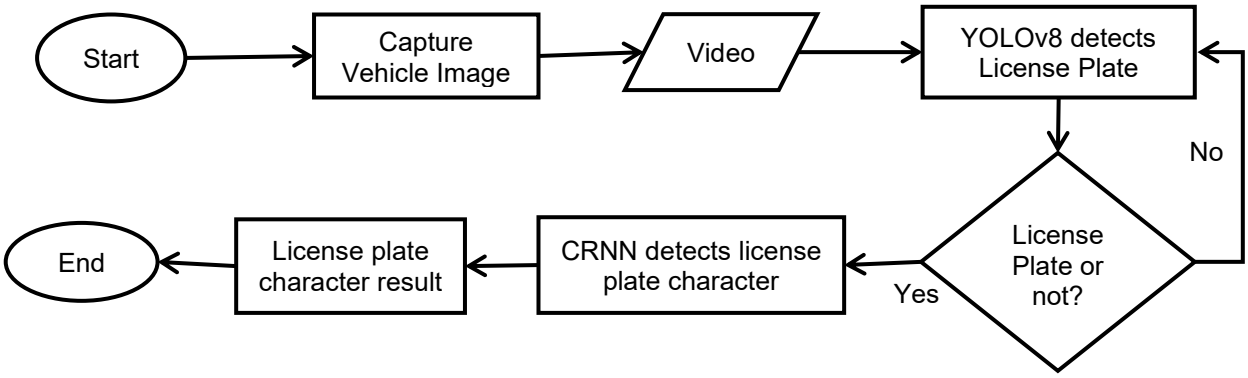


Figure 3. Prototype Workflow

The license plate detection model using YOLOv8 and the character recognition model using CRNN are installed and run on the Raspberry Pi. To obtain image input, the camera is connected to the Raspberry Pi to capture images or videos from the surrounding environment and send them to the Raspberry Pi to be processed by the CRNN model. Once the CRNN model successfully recognizes the license plate character from the image or video input, the character string representing the vehicle's license plate will be used as a parking entrance ticket.

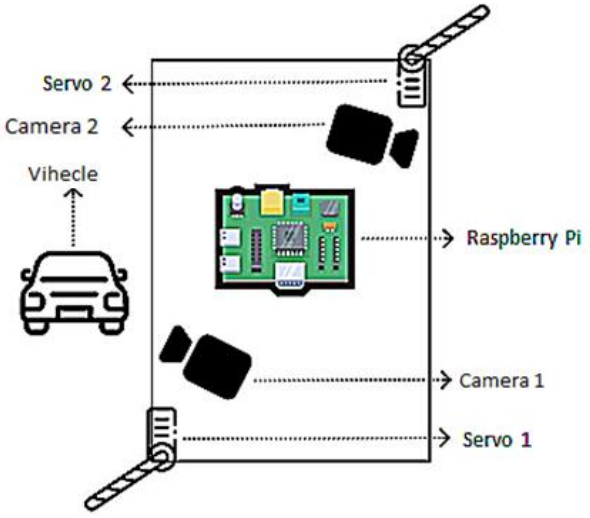


Figure 4. Prototype Design

Figure 4 displays how the prototype works. When a vehicle is detected at the entrance, the camera identifies the license plate area in the image using the YOLOv8 method. The area containing the license plate is then cropped and sent to the CRNN character recognition model to read and recognize the characters on the license plate. The resulting character string is processed by the Raspberry Pi as input to enter the parking lot. After successful recognition, the Raspberry Pi instructs the servo motor to open the barrier for the incoming vehicles. If the camera fails to detect a license plate, the servo motor will not open the entrance gate.

For the vehicle exit process, the camera again captures the vehicle license plate and checks whether the same plate was previously recorded during entry. If the license plate matches the store data, the servo motor opens the exit gate and the vehicle is allowed to leave. Conversely, if the vehicle license plate is not detected upon entry, the vehicle is not permitted to exit and the servo motor will not open the barrier.

4. Results and Discussion

4.1 YOLOv8 Training Results

The training results show a positive trend across the various matrices presented in Figure 5. The curves (box_loss, cls_loss, and dfl_loss) for the training and validation data showed a consistent decrease, indicating that the model successfully learned the patterns in the dataset. The precision and recall matrices show a steady increase, indicating the model's excellent ability to detect and classify license plates. Furthermore, the mAP50 and mAP50-95 graphs also show consistent improvement, reaching approximately 0.98 and 0.71 at the end of training, indicating strong detection performance under various IoU thresholds.

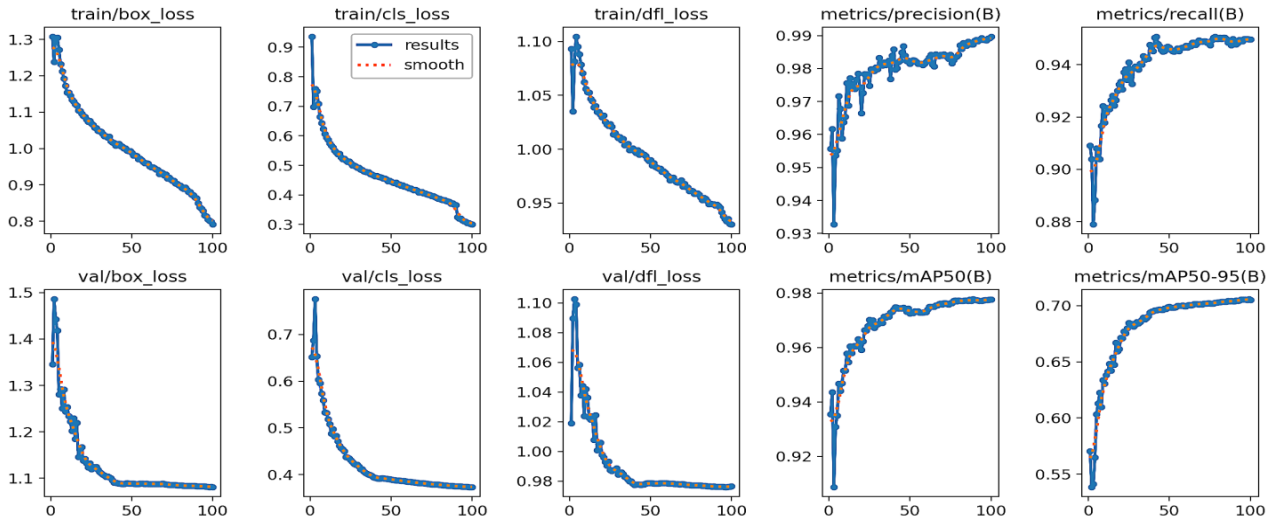


Figure 5. Learning Curve of YOLOv8

The label visualization in Figure 6 provides an overview of the characteristics of the license plate dataset used to train the YOLOv8 model. The bar graph in the upper-left corner shows that the dataset is dominated by a single class, namely “License_Plate,” with approximately 20,000 instances, indicating a high object focus. Meanwhile, the bounding box visualization in the upper-right corner illustrates the variation in license plate sizes and positions across the dataset. Most license plates are concentrated in the central area of the image and fall within the coordinate ranges x: 0.2-0.8 and y: 0.2-0.8, with varying sizes but relatively consistent aspect ratios (width: 0.1-0.4, height: 0.1-0.3). This visualization illustrates that the dataset has both diversity and consistency, making it ideal for training license plate detection models.

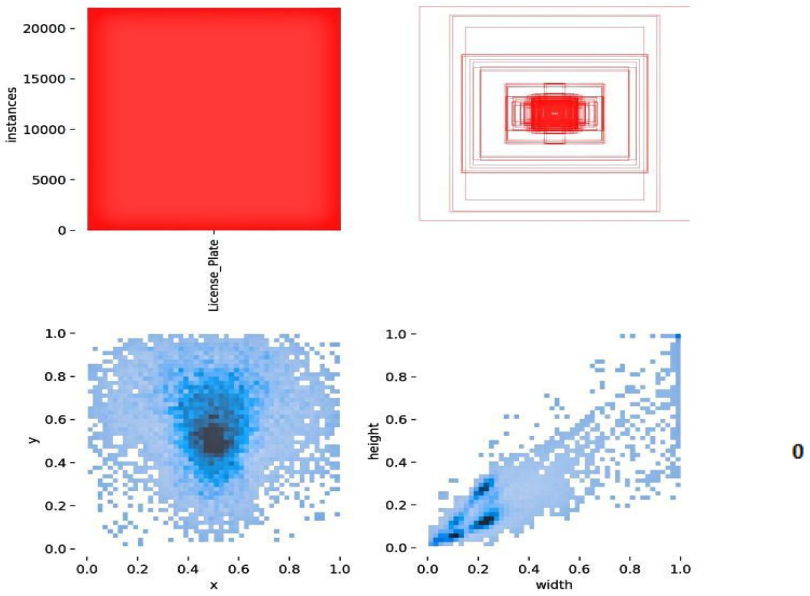


Figure 6. Visualization of License Plate Label

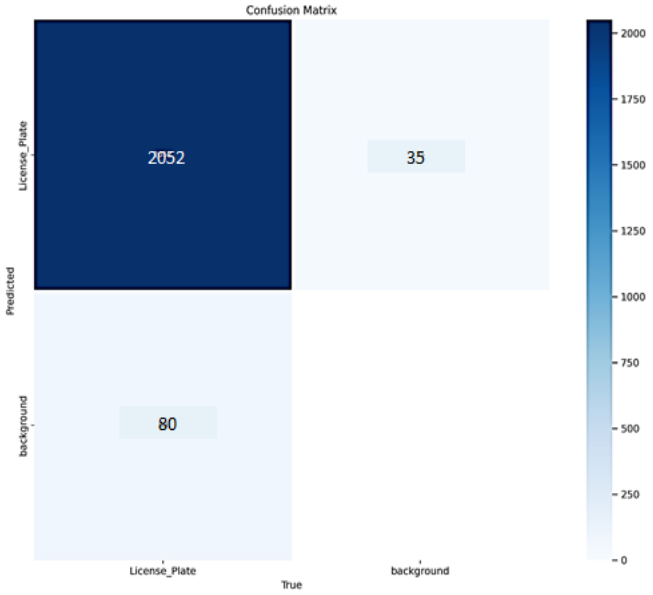


Figure 7. Confusion Matrix of Yolov8

4.2 YOLOv8 Model Evaluation Results

The evaluation of the trained YOLOv8 model for vehicle license plate detection shows a very impressive performance based on both the metrics and the visualization of the results. The confusion matrix shown in Figure 7 gives a clear picture of the model's accuracy in detecting vehicle license plate. The test results show that out of 2,167 total samples, the model produced 2,052 True Positives (TP), 35 False Positives (FP), and 80 False Negatives (FN), with 0 True Negatives (TN). Table 1 presents the calculated values for Accuracy, Precision, Recall, and F1-Score. As shown in Figure 8, which displays sample images of cars and motorcycles, the majority of license plates were successfully detected, with only a few instances of False Positives and False Negatives.

Accuracy is employed to evaluate the overall correctness of the model's predictions on the test dataset. A higher accuracy value indicates a better capability of the model in correctly detecting vehicle license plates, as shown in Equation 1.

$$Acurracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{1}$$

Precision, shown in Equation 2, represents the ratio of correctly detected license plates to the total positive predictions. Hence, a high precision value indicates that the model rarely misclassifies other objects as license plates.

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{2}$$

Recall, shown in Equation 3, evaluates the model's ability to detect all license plates actually present in the image. A high recall value indicates that only a small number of license plates are missed.

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3}$$

The F1-Score, shown in Equation 4, represents the harmonic mean of precision and recall, reflecting the balance between avoiding false detections and successfully identifying all license plates.

$$F1\ Score = 2\ \mathrm{x} \frac{\mathrm{precision\ x\ recall}}{\mathrm{precision} + \mathrm{recall}} \tag{4}$$

Table 1. Confusion Matrix Results

Model	Accuracy	Precision	Recall	F1 score
YOLOv8	94,69	98,32	96,25	97,27

Figure 8. Vehicle License Plate Prediction

4.3 CRNN training process

A Convolutional Recurrent Neural Network (CRNN) model for vehicle license plate identification was trained using an optimized hyperparameter configuration. The training parameters consisted of 100 epochs with batch size of 8, an input image size of 128×32 pixels, and the Adam optimizer with an adaptive learning rate. This relatively small batch size was selected to accommodate memory limitations on the Raspberry Pi device while increasing the model's generalization ability through more frequent parameter updates.

The training process of the CRNN model shows significant and consistent progress over 100 epochs, as illustrated in the training graph. In Figure 9, the training loss curve shows a rapid and steady decrease, starting at approximately 25 in the initial epoch and decreasing exponentially until it reaches a value below 5 by the end of training. This sharp decrease indicates that the model successfully recognizes the important patterns and features of the training dataset. Similarly, the validation loss follows the same downward trend, albeit with a larger change and a slightly higher final value than the training loss. This shows that the model not only memorizes the training data but is also able to generalize to unseen data.

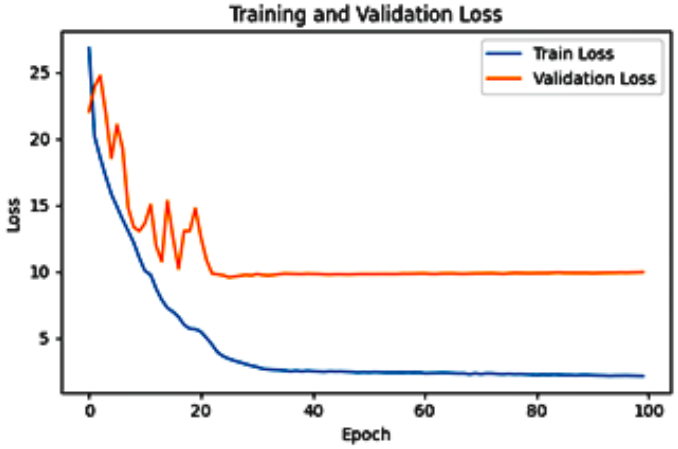


Figure 9. Training and Validation Loss

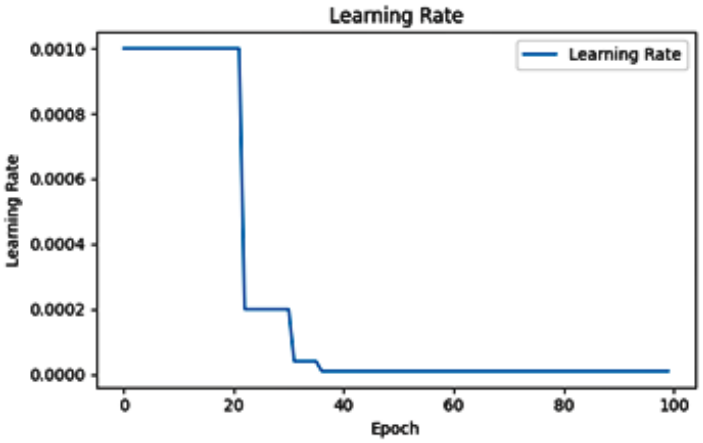


Figure 10. Learning Rate

Throughout the training process, the gap between the training loss and validation loss remains small and stable, indicating that the model does not experience significant overfitting. In Figure 10, the learning rate graph shows several decreases during training as a result of applying the ReduceLROnPlateau callback. Three significant decreases in the learning rate are observed: the first around the 20th epoch, the second around the 40th epoch, and the third around the 60th epoch. Each reduction is followed by an improvement in the validation loss, indicating that this adaptive strategy was effective in helping the model overcome plateau phases and achieve better performance.

4.4 Prototype Circuit

For the assembly of the vehicle license plate detection system, the components used include servo motors, cameras, and a Raspberry Pi. Based on the wiring diagram shown in Figure 11, the prototype circuit integrates the Raspberry Pi as the main processing unit with two servo actuators and two USB cameras. The servo motors are connected to the Raspberry Pi GPIO pins using three connections: the power pin (VCC) marked with a red wire is connected to the 5V pin, the ground pin (GND) marked with a black wire is connected to the ground pin, and the control signal pin (PWM) marked with a yellow wire is connected to GPIO pins 23 and 24. This configuration allows the Raspberry Pi to precisely control the servo motor position through PWM signals.

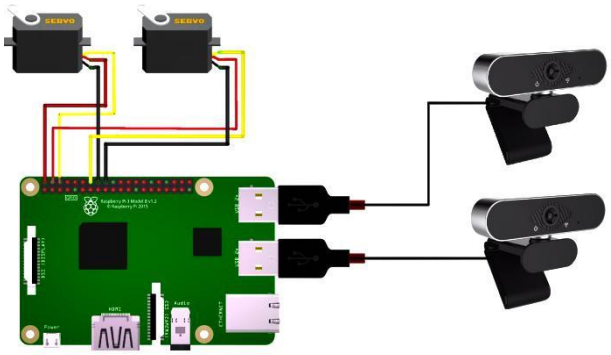


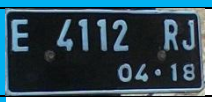
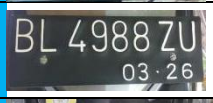
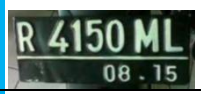
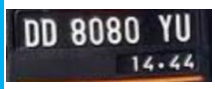
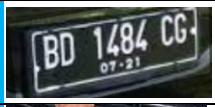
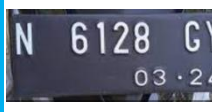
Figure 11. Prototype Circuit

In addition, two USB cameras were connected directly to the available USB ports on the Raspberry Pi using USB cables. This connection enables the transfer of visual data from the cameras to the Raspberry Pi for image processing using the YOLOv8 and CRNN algorithms implemented in the system. The use of two cameras along with two servos is designed to effectively control the entrance and exit bars.

4.5 Testing the Prototype with Raspberry Pi

In testing using Raspberry Pi, the first step was to connect the Raspberry Pi and the laptop to the same internet network using Advanced Ip Scanner. Once connected, the Raspberry Pi was accessed through RealVNC Viewer. License plate detection testing began with the entrance camera detecting the license plate area, followed by the CRNN model recognizing the characters on the license plate. Vehicle license plate detection testing at the entrance gate using the Raspberry Pi was carried out on 30 license plates samples, as shown in Table 2.

Table 2. Test Results of License Plate Detection at the Parking Lot Entrance

No	Plate Number	Prediction Result	Read/Total Characters Plate X 100	Result (%)	No	Plate Number	Prediction Result	Read/Total Characters Plate X 100	Result (%)
1.		E 4112 RJ	7/7 X 100	100%	16		BH 1375 NH	8/8 X 100	100%
2.		BL 4988 ZU	8/8 X 100	100%	17		BM 27 DR	6/6 X 100	100%
3.		AA 5356 HI	6/8 X 100	75%	18		H 2325 PI	7/7 X 100	100%
4.		KB 5179 FA	8/8 X 100	100%	19		R 4150 ML	7/7 X 100	100%
5.		I 6437 TE	5/7 X 100	75%	20		AB 4848 DH	8/8 X 100	100%
6.		BH 2618 MA	8/8 X 100	100%	21		L 1445 DZ	6/7 X 100	85,7%
7.		IB 5332 HM	7/8 X 100	87,5%	22		K 6362 TE	7/7 X 100	100%
8.		DD 8080 YU	8/8 X 100	100%	23		AA 4795 BE	8/8 X 100	100%
9.		BD 1484 CC	7/8 X 100	87,5%	24		H 3188 KK	7/7 X 100	100%
10.		BH 1956 MA	6/8 X 100	75%	25		B 6936 SK	7/7 X 100	100%
11.		BL 1 HC	5/5 X 100	100%	26		AD 4247 IW	8/8 X 100	100%
12.		DK 463 AH	7/7 X 100	100%	27		AD 4247 IW	8/8 X 100	100%
13.		E 5303 HL	7/7 X 100	100%	28		C 3540 DV	6/7 X 100	85,7%
14.		N 6128 GY	7/7 X 100	100%	29		C 5071 YV	6/7 X 100	85,7%
15.		H 6888 OL	5/7 X 100	71,4%	30		H 2019 HB	6/7 X 100	85,7%

Average	**93,8%**

The results showed that the CRNN achieved an average character recognition accuracy of 93.8%.

During prototype testing, the entrance camera detected the vehicle license plate and its characters to enter the parking lot. Once the license plate and its characters were successfully detected, the servo acting as the entrance barrier opened, allowing the vehicle to enter while displaying the day, date, month, and time of entry. Similarly, for the exit process, the camera re-detected the license plate previously recorded at the entrance. After detection, the servo opened the exit barrier and displayed the day, date, month, entry time, exit time, and the duration of the vehicle's stay in the parking lot. The results of license plate detection during vehicle entry and exit on the Raspberry Pi system are shown in Figure 12 and 13.

Figure 12. Detection Results on the Raspberry Pi Device

Figure 13. Prototype of Opening the Vehicle Entrance Bar

A review of the scientific literature shows that the license plate recognition systems and character detection methods have been the focus of many previous studies. To provide a deeper overview of developments in this field, Table 3 summarizes several relevant studies. This summary includes various approaches, methodologies, and results achieved by previous researchers, thus serving as a basis for comparison and validation of the current study.

Table 3. Comparison with Previous Research

Research	Method	Result
[6]Sugeng et al. (2023)	CRAFT & CRNN	85,73%
[4]Y. Elhadi et al. (2019)	Faster R-CNN & CNN	93%
[3]Abdellatif et al. (2023)	IoT with algorithm OCR	93%
This research	YOLOv8 & CRNN	93,8%

One study integrated CRAFT and CRNN models to detect and recognize text on vehicle license plates with for automatic parking system enhancement, achieving an accuracy of 85.73% [6]. Another study utilized Faster R-CNN for license plate detection, image processing algorithms for character segmentation, and a CNN model for character recognition, reaching an overall recognition accuracy of 93% [4]. On the other hand, a lightweight IoT-based license plate recognition system using the OCR algorithm successfully recognized Arabic license plates with an accuracy of up to 93% [3]. Furthermore, this study also developed a prototype license plate detection system employing YOLOv8 to localize license plate regions in vehicle images and CRNN for character recognition. The system achieved a character recognition accuracy of 93.8% and was successfully implemented on a Raspberry Pi-based prototype.

5. Conclusion

This study successfully designed and implemented an intelligent parking system based on automatic license plate recognition, leveraging the YOLOv8 model for license plate detection and a Convolutional Recurrent Neural Network (CRNN) for character recognition. The system was deployed on a Raspberry Pi platform, providing a low-cost, efficient, and real-time solution suitable for embedded and IoT-based applications. Experimental results demonstrate that the YOLOv8 model achieved strong performance, with a detection accuracy of 94.69%, precision of 98.32%, recall of 96.25%, and an F1-score of 97.27%. Concurrently, the CRNN model achieved an average character recognition accuracy of 93.8% across 30 tested license plates.

The prototype successfully managed vehicle access by automatically detecting license plates during both entry and exit events and accurately calculating parking duration based on recorded timestamps. However, despite the promising results, the system experienced some character recognition errors—particularly among visually similar characters such as misclassifying 'G' as 'C', 'Q' as 'O', and 'W' as 'H'. These findings highlight the need for further refinement in character segmentation and model robustness.

The primary contributions of this research include the development of a functional end-to-end smart parking prototype and the successful deployment of deep learning models on resource-constrained hardware. To further enhance system performance and broaden applicability, several research directions are recommended, including the comparison of alternative object detection models (e.g., YOLOv9, EfficientDet) and text recognition frameworks (e.g., Transformer-based OCR). Additionally, exploring hybrid model architectures and implementing more advanced character segmentation techniques could further improve recognition accuracy, particularly in complex or variable real-world environments.

References

[1] B. P. Statistik, “Perkembangan Jumlah Kendaraan Bermotor.”.
[2] R. Kanan and H. Arbess, “An IoT-Based Intelligent System for Real-Time Parking Monitoring and Automatic Billing,” *2020 IEEE Int. Conf. Informatics, IoT, Enabling Technol. ICIoT 2020*, pp. 622–626, 2020. https://doi.org/10.1109/ICIoT48696.2020.9089589
[3] M. M. Abdellatif, N. H. Elshabasy, A. E. Elashmawy, and M. AbdelRaheem, “A low cost IoT-based Arabic license plate recognition model for smart parking systems,” *Ain Shams Eng. J.*, vol. 14, no. 6, p. 102178, 2023. https://doi.org/10.1016/j.asej.2023.102178
[4] Y. Elhadi, O. Abdalshakour, and S. Babiker, “Arabic-numbers recognition system for car plates,” *Proc. Int. Conf. Comput. Control. Electr. Electron. Eng. 2019, ICCCEEE 2019*, no. September 2019, 2019. https://doi.org/10.1109/ICCCEEE46830.2019.9071288
[5] P. Choorat, C. Sirikornkarn, and T. Pramoun, “License Plate Detection and Integral Intensity Projection for Automatic Finding the Vacant of Car Parking Space,” *34th Int. Tech. Conf. Circuits/Systems, Comput. Commun. ITC-CSCC 2019*, pp. 4–7, 2019. https://doi.org/10.1109/ITC-CSCC.2019.8793297
[6] W. Sugeng and R. Husaini, “Implementasi Convolutional Recurrent Neural Network untuk Identifikasi Plat Nomor Mobil pada Sistem Parkir Otomatis,” *MIND (Multimedia Artif. Intell. Netw. Database) J.*, vol. 8, no. 2, pp. 142–157, 2023. https://doi.org/10.26760/mindjournal.v8i2.142-157
[7] A. Menon and B. Omman, “Detection and Recognition of Multiple License Plate from Still Images,” *2018 Int. Conf. Circuits Syst. Digit. Enterp. Technol. ICCSDET 2018*, pp. 1–5, 2018. https://doi.org/10.1109/ICCSDET.2018.8821138
[8] Y. M. Alwaqfi and M. Mohamad, “A review of Arabic optical character recognition techniques & performance,” *Int. J. Eng. Trends Technol.*, no. 1, pp. 44–51, 2020. https://doi.org/10.14445/22315381/CATI1P208

[9] G. Sharma, "Performance Analysis of Vehicle Number Plate Recognition System Using Template Matching Techniques," *J. Inf. Technol. Softw. Eng.*, vol. 08, no. 02, 2018. https://doi.org/10.4172/2165-7866.1000232
[10] G. Hajare, U. Kharche, P. Mahajan, and A. Shinde, "Automatic Number Plate Recognition System for Indian Number Plates using Machine Learning Techniques," *ITM Web Conf.*, vol. 44, p. 03044, 2022. https://doi.org/10.1051/itmconf/20224403044
[11] A. S., J. Yankey, and E. O., "An Automatic Number Plate Recognition System using OpenCV and Tesseract OCR Engine," *Int. J. Comput. Appl.*, vol. 180, no. 43, pp. 1–5, 2018. https://doi.org/10.5120/ijca2018917150
[12] F. N. M. Ariff, A. S. A. Nasir, H. Jaafar, and A. Zulkifli, "Sauvola and Niblack Techniques Analysis for Segmentation of Vehicle License Plate," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 864, no. 1, 2020. https://doi.org/10.1088/1757-899X/864/1/012136
[13] M. Hussain, "YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection," *Machines*, vol. 11, no. 7, 2023. https://doi.org/10.3390/machines11070677
[14] M. Talib, A. H. Y. Al-Noori, and J. Suad, "YOLOv8-CAB: Improved YOLOv8 for Real-time Object Detection," *Karbala Int. J. Mod. Sci.*, vol. 10, no. 1, pp. 56–68, 2024. https://doi.org/10.33640/2405-609X.3339
[15] M. Q. Yao Guangzhen, Sandong Zhu, Long Zhang, "HP-YOLOv8 : High-Precision Small Object Detection Algorithm For remote sensing Images," *Sensors*, no, pp. 1–23, 2024. https://doi.org/https://doi.org/10.3390/s24154858
[16] H. Lou *et al.*, "DC-YOLOv8: Small-Size Object Detection Algorithm Based on Camera Sensor," *Electron.*, vol. 12, no. 10, pp. 1–14, 2023. https://doi.org/10.3390/electronics12102323
[17] T. M. Al-Hasan, V. Bonnefille, and F. Bensaali, "Enhanced YOLOv8-Based System for Automatic Number Plate Recognition," *Technologies*, vol. 12, no. 9, pp. 1–26, 2024. https://doi.org/10.3390/technologies12090164
[18] Z. Hu, Y. Hu, J. Liu, B. Wu, D. Han, and T. Kurfess, "A CRNN module for hand pose estimation," *Neurocomputing*, vol. 333, pp. 157–168, 2019. https://doi.org/10.1016/j.neucom.2018.12.065
[19] A. Onan, "Bidirectional convolutional recurrent neural network architecture with group-wise enhancement mechanism for text sentiment classification," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 5, pp. 2098–2117, 2022. https://doi.org/10.1016/j.jksuci.2022.02.025
[20] X. Wang, W. Jiang, and Z. Luo, "Combination of convolutional and recurrent neural network for sentiment analysis of short texts," *COLING 2016 - 26th Int. Conf. Comput. Linguist. Proc. COLING 2016 Tech. Pap.*, pp. 2428–2437, 2016.
[21] A. H. Wenhua Yu, Mayire Ibrayim, "Scene Text Recognition Based on Improved CRNN," *Information*, no, pp. 1–14, 2023. https://doi.org/10.3390/info14070369
[22] Y. Liu, Y. Wang, and H. Shi, "A Convolutional Recurrent Neural-Network-Based Machine Learning for Scene Text Recognition Application," *Symmetry (Basel).*, vol. 15, no. 4, 2023. https://doi.org/10.3390/sym15040849
[23] B. Shi, X. Bai, and C. Yao, "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, 2017. https://doi.org/10.1109/TPAMI.2016.2646371
[24] S. Küçükdermenci, "Raspberry Pi-Based Real-time Parking Monitoring with Mobile App Integration," *5th Int. Conf. Eng. Appl. Nat. Sci. August 25-26, 2024 Konya, Turkey*, 2024.
[25] P. Devi, T. Sharanmai, and T. Chandrika, "IoT Based Smart Vehicle Parking and Automatic Billing System Using RFID," *J. Eng. Sci.*, vol. 14, no. 04, pp. 922–934, 2023.
[26] W. A. Jabbar, C. W. Wei, N. Atiqah, and A. M. Azmi, "Internet of Things An IoT Raspberry Pi-based parking management system for smart campus," *Fac. Electr. Electron. Eng. Technol. Univ. Malaysia Pahang, 26600 Pekan, Pahang, Malaysia*, p. 100387, 2021. https://doi.org/10.1016/j.iot.2021.100387
[27] A. O. Agbeyangi, O. A. Alashiri, and A. E. Otunuga, "Automatic Identification of Vehicle Plate Number using Raspberry Pi," *2020 Int. Conf. Math. Comput. Eng. Comput. Sci. ICMCECS 2020*, pp. 1–4, 2020. https://doi.org/10.1109/ICMCECS47690.2020.246983
[28] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai, "ASTER: An Attentional Scene Text Recognizer with Flexible Rectification," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–14, 2018. https://doi.org/10.1109/TPAMI.2018.2848939
[29] Roboflow, "License Plate Recognition Computer Vision Project,".
[30] Kaggle, "Indonesian Plate Number,".
[31] Roboflow, "Plat Nomor Image Dataset,".