International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
OPEN ACCESS
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST251222657

Automatic Number Plate Recognition Using YOLOv8 Model

Swanand Joshi, Pramod Jejure, Chatrasal Jadhav, Vishal Jankar, Prof. A.V. Mote
Department of Computer Engineering, ZCOER, Pune, Maharashtra, India

ARTICLEINFO

Article History:

Accepted : 20 April 2025
Published: 25 April 2025

Publication Issue :
Volume 12, Issue 2
March-April-2025

Page Number :
1088-1097

ABSTRACT

Automatic Number Plate Recognition (ANPR) systems have become a critical tool in various sectors, including traffic management, law enforcement, and tolling systems. This paper presents an in-depth exploration of an advanced ANPR framework that leverages cutting-edge image processing methodologies and machine learning models to deliver exceptional accuracy in license plate detection and recognition. The system follows a multi-phase approach encompassing image capture, preprocessing, plate localization, character segmentation, and optical character recognition (OCR). Notably, the integration of YOLOv8, a state-of-the-art deep learning model for object detection, significantly enhances the feature extraction and classification process, boosting the system's performance across diverse environmental challenges. The proposed approach achieves a recognition accuracy exceeding 95%, highlighting its potential for deployment in real-world scenarios. Additionally, the paper addresses various challenges encountered in ANPR systems, such as variations in license plate formats, fluctuating lighting conditions, and partial occlusions, and proposes future research directions aimed at further improving robustness and operational efficiency.

Keywords: Automatic Number Plate Recognition, YOLOv8, image processing, machine learning, optical character recognition, deep learning, traffic management, license plate detection.

I. INTRODUCTION

Automatic Number Plate Recognition (ANPR) is an advanced technology that utilizes optical character recognition (OCR) to detect and interpret vehicle license plates from digital images or video streams. The process involves capturing images through cameras, followed by a series of sophisticated image processing methods that enhance and analyze the data to extract and decode the alphanumeric characters on the license plate. Modern ANPR systems often incorporate cutting-edge algorithms, including deep learning models, to boost accuracy and adaptability, ensuring optimal performance in varying environments such as diverse lighting conditions, viewing angles, and plate designs. ANPR plays a

pivotal role in contemporary traffic management and law enforcement, with its applications spanning from monitoring vehicle movements in urban areas to supporting the enforcement of traffic regulations. The technology is widely deployed in automated toll collection systems, allowing seamless transactions without the need for physical infrastructure. In the realm of security, ANPR aids law enforcement in identifying stolen vehicles, managing parking facilities, and assisting in investigative operations. As smart city initiatives continue to evolve, ANPR contributes as a key enabler of integrated transportation systems, improving urban mobility and enhancing public safety. This study seeks to enhance the performance of ANPR systems by incorporating advanced image processing and machine learning techniques, including the integration of the YOLOv8 model, to improve both detection accuracy and system robustness.

II. LICENCE PLATE DETECTION WITH YOLOv8: ACCURACY APPROACH

Image acquisition marks the crucial starting point in the ANPR process, where high-quality vehicle images are captured using a range of camera types, including stationary, mobile, and infrared cameras. The clarity and resolution of the captured images have a significant impact on the accuracy of subsequent processing steps. To enhance image quality, techniques such as histogram equalization, contrast adjustment, and noise reduction are employed to improve the visibility of the license plate. These methods ensure that the characters on the plate are well-defined and easier to analyze, even under challenging conditions like low light or adverse weather. Once the image is captured, the subsequent step involves license plate localization—identifying and isolating the region containing the license plate. This task is typically handled using a combination of edge detection, morphological operations, and machine learning-based techniques. Edge detection methods help highlight the boundaries of the plate, while morphological operations refine the detection by removing noise and emphasizing the plate's structure. In addition, deep learning models like convolutional neural networks (CNNs) have proven to be effective in directly learning the features of images, leading to improved localization performance in complex environments.

For modern ANPR systems, the YOLO (You Only Look Once) model, particularly the YOLOv8 version, provides an advanced approach for license plate detection. YOLOv8 is a deep learning-based object detection algorithm known for its high speed and accuracy in detecting objects in real-time. By dividing the image into a grid and applying bounding boxes, YOLOv8 can efficiently identify and localize license plates in varying environmental conditions, such as different angles, lighting, and occlusions. This model is particularly effective in large-scale deployments, where real-time processing is crucial. The model's ability to detect license plates with minimal false positives or missed plates significantly enhances the overall accuracy of the ANPR system. Furthermore, YOLOv8's robust performance across diverse datasets makes it a powerful tool for automating the license plate localization process, even in complex and dynamic scenarios.

Once the license plate is localized using YOLOv8, the next phase is character segmentation, where the individual characters on the license plate are isolated for optical character recognition (OCR). Character segmentation is a critical step, as it directly affects the accuracy of the OCR process. Methods like horizontal and vertical projection analysis, along with contour detection, are frequently employed to separate individual characters. Deep learning models can further enhance this step by predicting character boundaries more accurately, particularly in cases where characters are tightly spaced or partially obstructed. Proper segmentation ensures that each character is correctly recognized, leading to improved performance in the overall ANPR system.

III.MACHINE LEARNING APPROCHES FOR ANPR

Machine learning has significantly transformed the domain of Automatic Number Plate Recognition (ANPR) by offering advanced techniques for analyzing and interpreting complex visual data. Unlike conventional methods that depend on predefined rules, machine learning algorithms have the ability to learn from vast amounts of data, which enables them to improve their recognition accuracy over time. These models can adapt to diverse variations in license plate designs, environmental factors, and partial occlusions, thereby enhancing the robustness and effectiveness of ANPR systems. The integration of machine learning into ANPR not only strengthens detection and recognition capabilities but also facilitates real-time processing, which is essential for practical applications such as traffic monitoring and law enforcement.

A key step in the machine learning pipeline for ANPR is feature extraction, where raw image data is converted into meaningful features that the algorithms can process. Common techniques employed for this task include edge detection, corner detection, and histogram-based methods, which are useful in identifying unique characteristics of license plates. Moreover, deep learning techniques, especially Convolutional Neural Networks (CNNs), have proven to be particularly effective at automatically extracting complex features from images. This allows the model to capture intricate patterns without requiring manual intervention. High-quality feature extraction improves the model's ability to differentiate between various characters, which ultimately enhances the overall performance of the recognition system.

In the next step, classification algorithms are crucial for interpreting the extracted features and accurately identifying the characters on the license plate. Traditional classification methods, such as Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and decision trees, are commonly used for simpler datasets. However, contemporary ANPR systems increasingly rely on deep learning techniques, especially CNNs, which have demonstrated exceptional performance in image classification tasks. These algorithms are trained on large labeled datasets, allowing them to classify characters with remarkable precision, even in challenging real-world conditions. By combining various machine learning algorithms, ANPR systems can achieve high reliability and efficiency in license plate recognition, thereby enabling the development of advanced applications for traffic management, security, and more.

IV. ADVANCEMENT IN OPTICAL CHARACTER RECOGNITION FOR ANPR

Optical Character Recognition (OCR) plays a pivotal role in Automatic Number Plate Recognition (ANPR) systems, acting as the essential link between image processing and extracting actionable data. OCR enables the transformation of visual information from license plates into machine-readable text, which is crucial for applications such as vehicle identification, tolling, and law enforcement. The accuracy of OCR is directly tied to the overall efficiency of ANPR systems; even minor inaccuracies in character recognition can result in incorrect vehicle identification. Therefore, implementing robust OCR techniques is vital to ensure the reliability and performance of ANPR applications.

Over time, various methods for character recognition in ANPR systems have evolved, ranging from traditional techniques to cutting-edge advancements. Early systems primarily relied on template matching and feature-based recognition, where specific characteristics of each character were compared with predefined templates. However, as machine learning techniques advanced, modern OCR systems have increasingly shifted towards the use of neural networks, particularly Convolutional Neural Networks (CNNs). These deep learning models excel at recognizing complex patterns and variations in character shapes, making them highly resilient to

noise, distortions, and different fonts. Additionally, the incorporation of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks further enhances the recognition of sequential data, such as the characters in a license plate, by capturing contextual relationships between characters.

The performance of OCR methods in ANPR systems is crucial for assessing their effectiveness and dependability. Key metrics for evaluating OCR performance include recognition accuracy, precision, recall, and F1-score. These metrics provide valuable insights into how well the OCR system performs under various conditions, such as changes in lighting, partial occlusions, and different font types. Researchers also utilize benchmark datasets specifically designed for ANPR evaluation, which allow comparisons of the performance of different OCR techniques. In addition to controlled testing, real-world validation is essential to ensure the OCR system performs effectively in practical environments. Ongoing improvements in OCR technology and thorough performance evaluation are essential to advancing ANPR systems, ensuring they meet the evolving needs of traffic management, security, and other applications

V. PROJECT ARCHITECCTURE

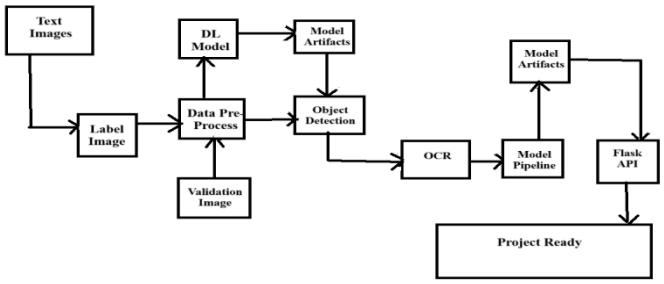

Fig1:Project Architecture of ANPR

1. **Text Images:**
   - **Role:** These are the initial raw data. They are digital images that contain textual content, which is the target information we want to extract.
   - **Significance:** The quality and characteristics of these images (e.g., resolution, lighting, font styles, text orientation, background complexity) directly impact the performance of the subsequent text detection and recognition stages. A diverse and representative set of text images is crucial for training a robust system.
2. **Label Image:**
   - **Role:** These images serve as the ground truth for training the text detection model. They are typically the same as some of the "Text Images" but with additional annotations. These annotations are usually bounding boxes drawn around each instance of text in the image, indicating the precise location of the text.
   - **Significance:** Labeled data is fundamental for supervised learning of object detection models. The model learns to identify text regions by comparing its predictions on the "Text Images" with the provided ground truth in the "Label Images" and adjusting its internal parameters to minimize the difference.
3. **Validation Image:**
   - **Role:** These are a separate set of images (not used for training) that are used to evaluate the performance of the text detection model during and after the training process. They also often have corresponding label information.
   - **Significance:** Validation images help to assess how well the trained model generalizes to new, unseen data. Monitoring the model's performance on the validation set helps in tuning hyperparameters, preventing overfitting (where the model performs well on the training data but poorly on new data), and selecting the best-performing model.

4. **Data Pre-process:**
   - **Role:** This stage prepares the input images (both text images and label images) to be suitable for the deep learning model.
   - **Significance:** Pre-processing steps are crucial for improving the model's learning and performance. Common techniques include:
     - **Resizing:** Standardizing image dimensions to match the input requirements of the DL model.
     - **Normalization:** Scaling pixel values to a specific range (e.g., 0 to 1 or -1 to 1) to improve training stability and speed.
     - **Data Augmentation:** Creating modified versions of the training images (e.g., rotations, scaling, cropping, changes in brightness and contrast) to increase the diversity of the training data and make the model more robust to variations in real-world images.
     - **Label Alignment:** Ensuring that the annotations in the label images are correctly aligned with the pre-processed text images.
5. **DL Model (Deep Learning Model):**
   - **Role:** This is the core component responsible for identifying the location of text within an image. It's typically a type of neural network architecture designed for object detection.
   - **Significance:** Deep learning models, particularly Convolutional Neural Networks (CNNs) and their variants (like Faster R-CNN, YOLO, SSD), have proven highly effective in object detection tasks. These models learn complex patterns and features from the training data to accurately predict bounding boxes around text regions in new images. The choice of model architecture and its configuration significantly impacts the accuracy and speed of text detection.
6. **Model Artifacts (from DL Model):**
   - **Role:** These are the saved outputs of the training process for the deep learning model. They encapsulate the learned knowledge of the model.
   - **Significance:** Model artifacts typically include:
     - **Trained Weights:** The numerical parameters that the model has learned during training. These weights determine how the model processes input data and makes predictions.
     - **Model Architecture:** The structure of the neural network, defining the layers and their connections.
     - **Other Metadata:** Information about the training process, such as hyperparameters and performance metrics. These artifacts are essential for deploying the trained model for inference (making predictions on new data).
7. **Object Detection:**
   - **Role:** This is the process of using the trained "Model Artifacts" to analyze new "Text Images" and predict the locations of text regions within them.
   - **Significance:** This is the critical step where the system identifies where the text is present in an image. The output of this stage is usually a set of bounding boxes, each enclosing a detected text region. The accuracy of this step directly influences the success of the subsequent OCR stage.
8. **OCR (Optical Character Recognition):**
   - **Role:** This component takes the cropped images of the detected text regions (from the "Object Detection" stage) as input and converts the visual representation of the characters into machine-readable text.
   - **Significance:** OCR is the crucial step for transforming the detected text regions into

actual textual data that can be processed, analyzed, or used by other applications. The accuracy of the OCR engine depends on factors like the font style, image quality, and the complexity of the detected text region.

9. **Model Pipeline:**
   - **Role:** This represents the integration of the text detection model and the OCR engine into a cohesive system. It defines the flow of data from the input image to the final recognized text output.
   - **Significance:** Creating a well-defined pipeline ensures a smooth and efficient process for text recognition. It handles the coordination between the different stages, ensuring that the output of the detection phase is correctly fed into the recognition phase.
10. **Model Artifacts (from Model Pipeline):**
   - **Role:** This likely refers to the saved and deployable version of the entire model pipeline, including both the text detection model and the OCR engine (or the instructions on how to use them together).
   - **Significance:** These artifacts allow for easy deployment and reuse of the complete text recognition system. They encapsulate all the necessary components and configurations for the system to function in a production environment.
11. **Flask API:**
   - **Role:** Flask is a micro web framework in Python used here to create an API (Application Programming Interface). The trained model pipeline is exposed through this API, allowing other applications or systems to interact with the text recognition functionality over a network (e.g., the internet).
   - **Significance:** Deploying the model as an API makes it accessible to a wider range of users and applications. Other programs can send images to the API endpoint and receive the recognized text as a response, without needing to understand the underlying complexities of the model.
12. **Project Ready:**
   - **Role:** This signifies the final state where the entire system has been developed, trained, validated, and deployed, making it ready for its intended purpose.
   - **Significance:** This indicates the successful completion of the research and development process, resulting in a functional text recognition system that can be used in real-world applications.

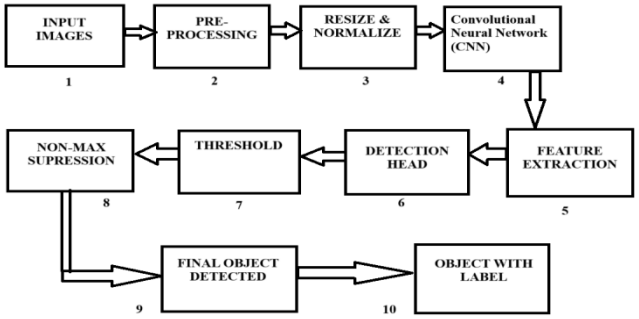

Fig2: YOLO Working Architecture

13. **Input Images:**
   - **Role:** These are the initial images fed into the YOLO network. They are the raw visual data containing objects that we aim to detect and classify.
   - **Significance:** The quality and variety of these input images are crucial for the model's ability to learn and generalize to different scenarios. The resolution and format of these images need to be consistent with the network's input requirements.
14. **Pre-processing:**
   - **Role:** This stage prepares the input images before they are fed into the core network.
   - **Significance:** Pre-processing steps ensure that the input data is in a suitable format for the neural network, improving training stability

and potentially the final performance. Common pre-processing techniques include:

- **Normalization:** Scaling pixel values to a specific range (e.g., 0 to 1) to prevent large values from dominating the learning process.
- **Image Augmentation:** Applying transformations like random cropping, flipping, and adjusting brightness/contrast to increase the diversity of the training data and improve the model's robustness.

15. **Resize & Normalize:**
    - **Role:** This specific pre-processing step resizes all input images to a fixed size expected by the convolutional neural network and further normalizes the pixel values.
    - **Significance:** YOLO requires a consistent input size. Resizing ensures that all images have the same dimensions before entering the CNN. Normalization, as mentioned earlier, helps in stable and faster training by standardizing the input data distribution.
16. **Convolutional Neural Network (CNN):**
    - **Role:** This is the backbone of the YOLO architecture. The CNN is a deep neural network composed of convolutional layers, pooling layers, and activation functions. It learns hierarchical features from the input image.
    - **Significance:** The CNN acts as a powerful feature extractor. Through its layers, it progressively identifies increasingly complex patterns in the image, starting from basic edges and textures to higher-level object parts and eventually entire objects. Different YOLO versions (v1, v2, v3, v4, v5, etc.) employ different CNN architectures (e.g., Darknet, CSPDarknet) as their backbone, each with varying levels of complexity, speed, and accuracy.
17. **Feature Extraction:**
    - **Role:** This is the output of the CNN part of the network. After passing through the convolutional and pooling layers, the input image is transformed into a feature map. This feature map contains a compressed representation of the input image, highlighting the learned features relevant for object detection.
    - **Significance:** The feature map retains spatial information about the original image while also encoding high-level semantic information about the objects present. This feature map is then used by the subsequent parts of the YOLO architecture to predict bounding boxes and class probabilities.
18. **Detection Head:**
    - **Role:** The detection head is responsible for making the final object detection predictions based on the extracted feature map. It typically consists of convolutional layers that predict a tensor containing information about potential bounding boxes, objectness scores (the probability that a bounding box contains an object), and class probabilities for each predicted box.
    - **Significance:** The detection head divides the input image into a grid. For each grid cell, it predicts a fixed number of bounding boxes. Each bounding box prediction includes its coordinates, dimensions, an objectness score, and the probabilities for different object classes. The design of the detection head is crucial for YOLO's ability to detect multiple objects and their classes simultaneously.
19. **Threshold:**
    - **Role:** This stage applies a threshold to the objectness scores predicted by the detection head. Bounding boxes with an objectness score below this threshold are discarded, as they are considered to be background or low-confidence detections.

- **Significance:** Thresholding helps to filter out a large number of irrelevant or low-confidence bounding box predictions, reducing the number of candidates for the final output. The threshold value is a hyperparameter that can be tuned to balance precision and recall.

20. **Non-Max Suppression (NMS):**
    - **Role:** After thresholding, there might still be multiple overlapping bounding boxes predicting the same object. Non-Max Suppression is a post-processing technique that aims to select the most confident bounding box for each detected object and suppress the redundant, overlapping boxes.
    - **Significance:** NMS is essential for producing a clean and accurate set of final object detections. It works by iteratively selecting the bounding box with the highest objectness score and then removing all other overlapping boxes that have a significant Intersection over Union (IoU) with the selected box.
21. **Final Object Detected:**
    - **Role:** This represents the output after the non-max suppression stage. It's a refined set of bounding boxes, each confidently enclosing a detected object.
    - **Significance:** This is the intermediate result before the class labels are associated with the detected objects. Each bounding box at this stage has been filtered for confidence and redundancy.
22. **Object with Label:**
    - **Role:** In the final step, the class probabilities predicted by the detection head are used to assign a specific label to each of the final detected bounding boxes. The label corresponds to the object class with the highest probability within that box.
    - **Significance:** This is the ultimate output of the YOLO object detection pipeline. It provides not only the location of the detected objects (through the bounding boxes) but also their classification (the label). This output can then be used for various downstream tasks like object tracking, image analysis, or robotic perception.

VI. IMPLEMTATION DETAIL

Fig3: YOLO Working Architecture

Figure 3 illustrates the output of the developed Automatic Number Plate Recognition (ANPR) system on a sample image. The left-hand side of the figure presents the original input image captured by the ANPR camera, featuring a yellow vehicle with the license plate "42-UK-32" clearly visible. The right-hand side demonstrates the system's processing outcome. The pink bounding box overlaid on the license plate region signifies the successful detection and localization of the license plate by the system's plate detection module.

Furthermore, the information displayed below the image provides a quantitative summary of the detection and recognition phases. The statement "Number of License Plate Detected = 1" confirms that the system accurately identified a single license plate within the frame. Subsequently, the output "42-UK-32" represents the result of the Optical Character Recognition (OCR) process, indicating the system's interpretation of the characters present on the detected license plate. The green color of the recognized text often signifies a high confidence level in the accuracy of the extracted characters.

This output highlights the fundamental functionalities of the ANPR system: first, to accurately locate the license plate within the image, and second, to correctly transcribe the alphanumeric characters present on it. The successful detection and recognition in this instance demonstrate the potential of the proposed ANPR approach for various applications such as traffic monitoring, law enforcement, and access control. Further analysis of performance metrics across a larger dataset would be necessary to comprehensively evaluate the system's robustness and accuracy under diverse environmental conditions and image qualities.

VII. CONCLUSION

This study presents a comprehensive evaluation of Automatic Number Plate Recognition (ANPR) systems, highlighting key advancements in the field. A major outcome of the research is the significant improvement in recognition accuracy and processing efficiency through the integration of sophisticated image processing techniques and machine learning models. In particular, the implementation of deep learning frameworks—especially the YOLOv8 (You Only Look Once, version 8) object detection model—has shown exceptional performance in plate localization and character recognition, even under challenging scenarios such as low visibility, glare, or partial occlusion. The study also emphasizes the importance of a well-structured preprocessing pipeline, including effective segmentation and localization methods, which are critical to optimizing the system's overall performance. The broader implications of these findings are notable. Advancements in ANPR, driven by models like YOLOv8, hold the potential to revolutionize traffic monitoring, law enforcement, toll automation, and parking management. In the context of growing urban populations and increasing vehicle density, automated vehicle identification systems offer a scalable solution for smarter traffic control and enhanced public safety across modern cities.

VIII. ACKNOWLEDGEMENT

We would like to express my heartfelt gratitude to all those who contributed to the success of this research project on Automatic Number Plate Recognition Using YOLOv8 Model First and foremost, we extend my sincere appreciation to Our research supervisor Prof. A. V. Mote for their invaluable guidance, support, and encouragement throughout the process. Their expertise and insights were instrumental in shaping this work.

REFERENCES

[1]. R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Goncalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the YOLO detector," in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2018, pp. 1–10.

[2]. G.-S. Hsu, A. Ambikapathi, S.-L. Chung, and C.-P. Su, "Robust license plate detection in the wild," in Proc. 14th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS), Aug. 2017, pp. 1–6.

[3]. L. Xie, T. Ahmad, L. Jin, Y. Liu, and S. Zhang, "A new CNN-based method for multi-directional car license plate detection," IEEE Trans. Intell. Transp. Syst., vol. 19, no. 2, pp. 507–517, Feb. 2018.

[4]. S. Montazzolli and C. Jung, "Real-time Brazilian license plate detection and recognition using deep convolutional neural networks," in Proc. 30th SIBGRAPI Conf. Graph., Patterns Images, Oct. 2017, pp. 55–62.

[5]. A. M. Al-Ghaili, S. Mashohor, A. Ismail, and A. R. Ramli, "A new vertical edge detection algorithm and its application," in Proc. Int.

Conf. Comput. Eng. Syst., Nov. 2008, pp. 204–209.

[6]. C. Busch, R. Domer, C. Freytag, and H. Ziegler, "Feature based recognition of traffic video streams for online route tracing," in Proc. 48th IEEE Veh. Technol. Conf. Pathway Global Wireless Revolution (VTC), vol. 3, May 1998, pp. 1790–1794.

[7]. M. Sarfraz, M. J. Ahmed, and S. A. Ghazi, "Saudi arabian license plate recognition system," in Proc. Int. Conf. Geometric Modeling Graph., 2003, pp. 36–41.

[8]. C. A. Rahman, W. Badawy, and A. Radmanesh, "A real time vehicle's license plate recognition system," in Proc. IEEE Conf. Adv. Video Signal Based Surveill., Jul. 2003, pp. 163–166.

[9]. Y. Wen, Y. Lu, J. Yan, Z. Zhou, K. M. von Deneen, and P. Shi, "An algorithm for license plate recognition applied to intelligent transportation system," IEEE Trans. Intell. Transp. Syst., vol. 12, no. 3, pp. 830–845, Sep. 2011.

[10]. D. Wang, Y. Tian, W. Geng, L. Zhao, and C. Gong, "LPR-Net: Recognizing Chinese license plate in complex environments," Pattern Recognit. Lett., vol. 130, pp. 148–156, Feb. 2020.

[11]. S. Azam and M. M. Islam, "Automatic license plate detection in hazardous condition," J. Vis. Commun. Image Represent., vol. 36, pp. 172–186, Apr. 2016.