# Methods for Automatic License Plate Recognition

Ho Sy Hieu - 20224271, Pham Huu Cuong - 20224303
ET-E16 K67

## Abstract

*The intelligent transportation system relies heavily on automatic license plate recognition (ALPR) for applications like traffic control, security, toll payment, and parking management. While many algorithms have been developed for license plate detection and recognition, they often face challenges under varying conditions. The rise of deep learning has significantly advanced computer vision, making ALPR a robust tool for automating traffic systems, parking management, and police surveillance. Despite its maturity, ALPR still requires optimization to achieve perfection. Recent years have witnessed significant progress in methodologies and performance in this field.*

*This work proposes an Automatic License Plate Detection and Recognition (ALPDR) system using YOLOv8 for license plate detection and PaddleOCR for recognition. The system is divided into three steps: License Plate Extraction, Character Segmentation, and Character Recognition. YOLOv8 ensures accurate and efficient license plate detection, while PaddleOCR facilitates robust character recognition. By integrating these advanced tools, this approach seeks to enhance the accuracy and efficiency of ALPDR systems, paving the way for further innovations in intelligent transportation systems.*

## 1. Introduction

Urban areas face growing challenges from increased vehicular traffic, including congestion, traffic violations, and vehicle theft. Automatic License Plate Detection and Recognition (ALPDR) is a vital component of intelligent transportation systems, leveraging image processing and artificial intelligence to identify vehicles by their license plates.

Traditional ALPDR systems involve three phases: preprocessing, license plate localization, and character recognition. However, they face challenges such as illumination changes, noise, motion blur, and diverse font styles and sizes. To address these issues, we propose a novel ALPDR system integrating YOLOv8 for license plate detection and PaddleOCR for character recognition. This combination enhances accuracy and efficiency while tackling complex scenarios such as varying lighting conditions and intricate backgrounds.

The objectives of this paper are:

- To present an ALPDR system using YOLOv8 and PaddleOCR.

- To improve detection and recognition accuracy with advanced techniques.

- To evaluate and compare the system against existing methods.

The paper is structured as follows: Section 2 reviews the literature; Section 3 outlines the proposed method; Section 4 presents results and analysis; and Section 5 concludes the study.

## 2. Literature survey

There are many frameworks that have been proposed for license plate detection and recognition, based on different image processing methods and artificial intelligence techniques. X. Ascar et al. [1] proposed the license plate detection technique which is based on kernel density function and binary techniques used for processing the license plate. The location of the license plate is what we found by multiplying the binary value and the original value of the image. Later on, the filtered binary value of the image has been used.

Ravi Kiran et al. [2] presented an image processing technique for Indian license plate detection and recognition with various conditions such as noisy environment, low light, non-standard license plate and cross angled situation. For the pre-processing they used several techniques such as gaussian smoothing, morphological transform and gaussian thresholding. For the segmentation they used contours and K-nearest neighbor algorithm for character recognition. Another method for Indian license plates has been proposed by Hanit Karwal et al. [3]. They address the problem of recognition of position of character and the scaling problem.

Fei Xie et al. [4] proposed a license plate detection and character recognition method based on a hybrid model composed of an effective feature extraction technique coupled with a Back-Propagation Neural Network (BPNN) classi-

fier. The author claims that their method can be able to overcome the problem of low illumination and complicated backgrounds. The pre-processing step has been taken to strengthen the image of the car and after locating the license plate on the car a feature extraction model is designed and finally, using backpropagation neural network model the character recognition is achieved.

In the proposed automatic license plate recognition detection for the steps has been proposed which includes license plate-extraction image preprocessing, character segmentation and character recognition. For each step you need a novel method and technique that has been applied at last for character recognition. The proposed CNN method has been used and compared the implementation result with different character recognition techniques.

## 3. Proposed Methods

We approach the problem of license plate recognition using two distinct methods to highlight the trade-offs between simplicity and advanced techniques. The first method relies on traditional image processing, including edge detection, contour analysis, and optical character recognition (OCR), providing a straightforward and computationally efficient solution. The second method employs a more sophisticated approach using YOLOv8 for license plate detection and PaddleOCR for character recognition, leveraging state-of-the-art deep learning techniques for improved accuracy and robustness in diverse conditions.

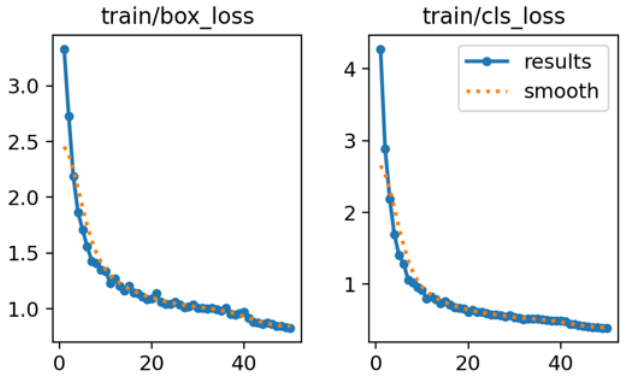
### 3.1. First method: Deep learning

Deep learning methods have revolutionized license plate detection and recognition with their ability to learn complex patterns and generalize across diverse conditions. In this approach, YOLOv8, a state-of-the-art object detection model, is utilized to accurately localize license plates within an image by leveraging its advanced convolutional neural network architecture. Once the license plate region is detected, PaddleOCR, a powerful optical character recognition framework, processes the extracted region to recognize the alphanumeric characters on the plate. This combination of YOLOv8 for detection and PaddleOCR for character recognition ensures high accuracy and robustness, even in challenging scenarios like varying lighting conditions, motion blur, or diverse license plate formats.

#### 3.1.1 Detection Training

Using a previously collected dataset from Roboflow, we trained a YOLOv8 deep learning model for license plate detection. This dataset, consisting of a diverse set of images featuring vehicles with varying license plate styles, backgrounds, and lighting conditions, provided a solid foundation for training a robust detection model. We utilized the YOLOv8 architecture due to its efficiency and high performance in object detection tasks. The model was trained over 50 epochs, and during this period, it learned to accurately detect and localize license plates in images.

During the training process, we observed distinct trends in both training and validation losses. In the early epochs, training losses, including bounding box (box loss), class (cls loss), and distribution focal loss (dfl loss), were relatively high. However, as the model progressed and adjusted its weights, these losses gradually decreased, indicating that the model was improving its ability to detect license plates. A similar trend was seen in the validation loss, which also showed a steady decline over time. This reduction in validation loss suggests that the model was generalizing well to unseen data and was not overfitting to the training set.




The precision values in the early stages of the training process were relatively low, indicating that the model was making some incorrect predictions and had a higher rate of false positives. However, as the training progressed and the model refined its weights and learned more from the dataset, there was a noticeable upward trend in precision. Over time, this improvement continued, with the model becoming increasingly effective at distinguishing between true license plate regions and irrelevant parts of the image. By the final epochs of training, the precision approached an impressive 99 percent, which is a clear indication that the model was correctly identifying license plates with very few false positives. This level of precision suggests that the model had learned to focus on the critical features of license plates and accurately classify them without mistakenly detecting non-plate regions as plates.

In parallel, the recall metric, which measures the model's ability to correctly identify all relevant license plate regions, also showed significant improvements. Initially, recall was not as high as desired, but as the model continued to train and adjust its parameters, recall steadily increased. This was particularly noticeable after the first few epochs, as the model began to adapt and better generalize to the

dataset. By the 50th epoch, recall remained consistently above 99 percent, demonstrating that the model was successfully identifying almost all of the true license plate regions in the images, with very few missed detections. A recall rate this high indicates that the model was highly effective at identifying the presence of license plates, leaving very few instances where a plate went undetected.

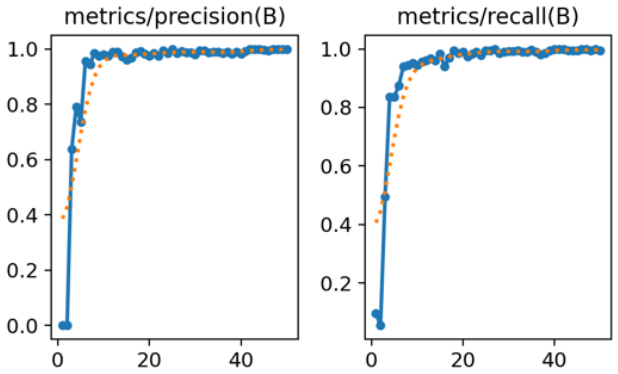



Figure 1. Precision and Recall

In conclusion, the training process led to significant improvements in the model's performance, with both precision and recall reaching over 99 percent by the final epochs. This indicates the model's ability to accurately identify license plates with minimal false positives and negatives. The steady enhancement of these metrics demonstrates the model's robustness and effectiveness, making it well-suited for real-world applications in automatic license plate recognition (ALPR) systems.

### 3.1.2 Plate Detection

The plate detection process begins by loading the trained YOLO model from the specified path and reading the input image using OpenCV. The image is then displayed using Matplotlib with color conversion to RGB for proper visualization. After the image is displayed, the model is applied to the image, generating predictions on potential regions containing license plates. The results from the YOLO model contain bounding boxes for each detected plate, which are iterated over to extract their coordinates. For each bounding box, the coordinates (x1, y1, x2, y2) are extracted and converted to integers. A green rectangle is drawn around the detected license plate region using OpenCV's rectangle function, highlighting the detected area. Additionally, a region of interest (ROI) corresponding to the detected plate is cropped from the image and stored in a list for further processing. This list of ROIs holds the localized license plates, which are then displayed individually using Matplotlib. Each cropped license plate region is shown sequentially for inspection. This process ensures that the license plate is accurately detected and localized within the image, preparing it for further recognition steps.


Figure 2. Detected/ Cropped Region

### 3.1.3 Character recognition

In this step, the PaddleOCR model is used to perform Optical Character Recognition (OCR) on the detected license plate regions (ROI). The model is initialized, which allows it to detect text at various angles, making it more robust to tilted license plates. The OCR process is applied to each region of interest (ROI) extracted during the plate detection phase. The results from the OCR model contain the detected text along with a confidence score for each line of text. The output is iterated over, and for each detected line, the detected text and its corresponding confidence score are printed. This confidence score indicates how confident the model is in the accuracy of the detected text, helping assess the reliability of the OCR results. The output provides both the textual information (such as the license plate number) and the model's confidence, which is useful for post-processing or validation in further stages of the license plate recognition system.


Figure 3. Result

In conclusion, the deep learning-based license plate detection and recognition method utilizing YOLO for object detection and PaddleOCR for optical character recognition proves to be an efficient and robust solution. YOLO effectively detects license plate regions within images, even in

complex conditions, by localizing and isolating the plate from the background. PaddleOCR then extracts the text from these regions, providing high accuracy with confidence scores that help assess the reliability of the recognition. This method not only achieves high precision and recall rates but also demonstrates robustness to variations in plate positioning and text orientation. Overall, this deep learning approach significantly enhances the accuracy and efficiency of license plate recognition systems, making it suitable for various real-world applications.

## 3.2. Another approach: Simple algorithm

Another method we propose involves image processing, contour detection, and optical character recognition.

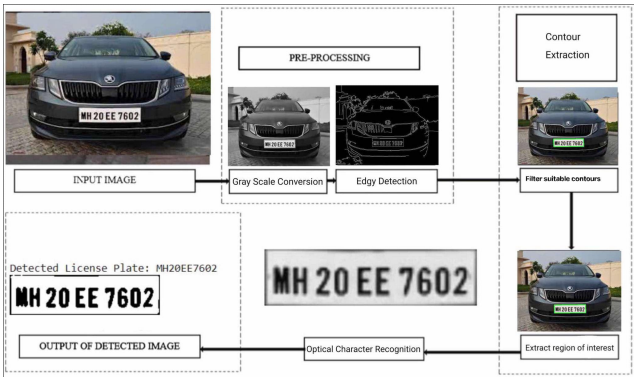



Figure 4. Flow chart of the process

### 3.2.1 Image Preprocessing

The preprocessing stage plays a critical role in preparing the image for the subsequent steps of the license plate detection and recognition process. It is aimed at enhancing the key features of the image while simultaneously reducing noise and irrelevant details, thus improving the accuracy of the overall model. The first step in preprocessing involves converting the input image to grayscale. By removing the color information, grayscale conversion allows the algorithm to focus solely on the intensity variations across the image. Next, a bilateral filter is applied to the grayscale image. This filter serves to smooth the image by reducing noise, while also preserving the sharpness of the edges. This is crucial for the subsequent steps where precise detection of objects, such as license plates, is required. By combining grayscale conversion with noise reduction and edge preservation, the preprocessing stage enhances the overall quality of the input image.

### 3.2.2 Edge Detection

Edges in the image are highlighted using the Canny edge detection method, which identifies areas of rapid intensity

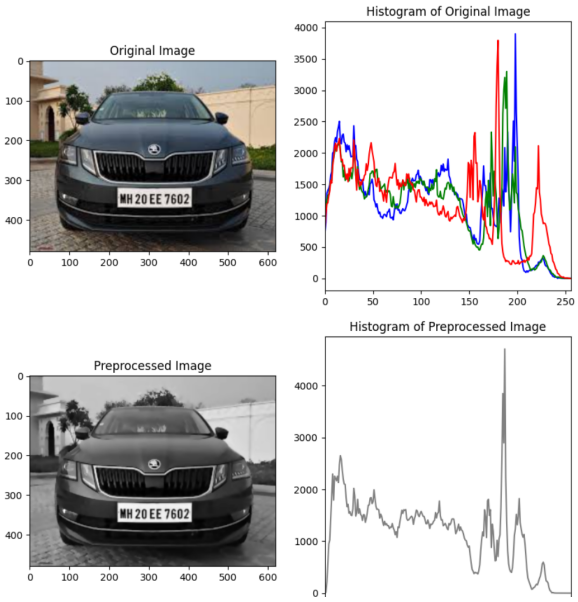



Figure 5. Image conversion

change. The parameters for edge detection are adjusted to optimize results.

It highlights areas of significant intensity changes in an image, making it easier to identify boundaries and structures. For license plate recognition, edge detection is used to isolate the edges of the license plate and its characters from the background. By transforming the image into a binary format where edges are represented as white lines on a black background, edge detection simplifies the task of identifying contours, enabling the subsequent steps of contour analysis and character segmentation.

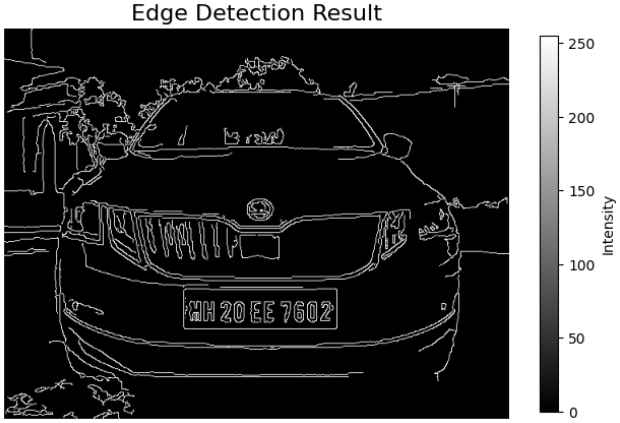



Figure 6. Detect edges from image

### 3.3. Detection Algorithm

This algorithm for license detection uses contour analysis on a processed image to isolate the region corresponding to the license plate. Initially, edge detection is performed on the input image to highlight significant intensity changes, producing a binary image where edges are represented as white lines. Contours are then identified from this binary image. The contours are sorted by area in descending order to prioritize larger regions. For each contour, the perimeter is calculated using `cv2.arcLength`, and an approximate polygon is generated using `cv2.approxPolyDP`. The algorithm selects the first contour that approximates a quadrilateral (indicative of a license plate's rectangular shape) and meets a minimum size threshold, ensuring only relevant regions are considered. If such a contour is found, it is drawn on the original image for visualization, and a mask is created to isolate the license plate region. The mask is applied to extract the corresponding region from the grayscale image, resulting in a cropped image of the license plate for further processing. If no suitable contour is found, the algorithm reports that no license plate was detected.

#### 3.3.1 Contour Detection and Approximation

Contour detection and approximation is a crucial step in the process of identifying regions of interest within an image, particularly for structured objects like license plates. The process begins with extracting contours, which represent the boundaries of objects in the image. This is achieved using the `cv2.findContours` function, which analyzes the binary edge-detected image to identify continuous curves that outline distinct regions. These contours are then retrieved using `imutils.grab_contours`, simplifying their handling and processing.

To focus on the most relevant contours, the algorithm sorts them in descending order of their area using `cv2.contourArea`. This prioritization ensures that larger regions, which are more likely to correspond to a license plate, are evaluated first. For each contour, the perimeter is computed using the `cv2.arcLength` function. The perimeter provides a measure of the contour's total boundary length, which is then used to approximate its shape.

The approximation is carried out using the `cv2.approxPolyDP` function, which simplifies the contour by reducing the number of points while preserving its overall structure. This method uses the Douglas-Peucker algorithm, controlled by a precision parameter that determines how closely the approximation matches the original contour. The algorithm checks if the approximated contour forms a quadrilateral by verifying that it has exactly four vertices. Additionally, it ensures that the contour's area exceeds a predefined threshold (`min_size`), filtering out smaller, irrelevant regions.

Contours meeting these criteria are considered potential candidates for a license plate and are stored for further processing. If no suitable contour is found, the algorithm concludes that no plate is detected. This meticulous process enables the identification of regions that are geometrically and dimensionally consistent with the expected appearance of a license plate, making it a foundational step in the overall detection pipeline.

#### 3.3.2 Plate Localization

Once a quadrilateral-shaped contour is found, it is assumed to be the license plate, given its similarity to the rectangular structure of most plates. The algorithm creates a binary mask corresponding to the detected contour, effectively isolating the identified region from the rest of the image. Using this mask, a bitwise operation, `cv2.bitwise_and`, is applied to the original grayscale image, suppressing irrelevant areas and highlighting the license plate.

The bounding coordinates of the contour are then calculated, allowing the algorithm to determine the precise location of the license plate within the image. These coordinates are used to crop the grayscale image, resulting in a focused and refined view of the license plate region. This cropped image serves as the localized license plate and is prepared for further stages, such as optical character recognition. If no suitable contour is detected, the algorithm concludes that no license plate is present, ensuring robustness in identifying valid regions.

### 3.4. Character Recognition

To optimize the optical character recognition (OCR) process, the localized license plate region undergoes a thresholding operation. The thresholding is applied using the Otsu's binarization method with OpenCV's `cv2.threshold` function. This step converts the cropped grayscale license plate image into a binary image, ensuring that the characters are distinctly visible against the background. The result is a high-contrast representation of the plate, which is ideal for text recognition.


Figure 7. Image binarization

The thresholded image is then processed using the Tesseract OCR engine with a specific page segmentation mode (`--psm 7`), which is tailored for single-line text

recognition. This configuration enables the accurate extraction of alphanumeric characters from the binary license plate image. The output from Tesseract includes the detected text, which may contain extraneous or invalid characters.

To refine the results, the detected text is passed through a fine-tuning function. This function filters out any characters that do not belong to a predefined set of valid license plate characters, including numbers and uppercase letters commonly used in license plates. The fine-tuned result represents the final detected license plate text. The thresholded binary plate image is also displayed for verification, ensuring that the preprocessing step produced a clear and usable input for the OCR engine.


Figure 8. Character recognition

By combining thresholding, OCR, and fine-tuning, this approach ensures robust character recognition, transforming the visual license plate into a machine-readable format.

### 3.5. Downsides of the Method

Although the described method for license plate detection and recognition is simple and efficient, it has several limitations:

- **Inaccuracy in Complex Scenarios:** The method relies on edge detection and contour analysis, which can fail under challenging conditions such as poor lighting, motion blur, or complex backgrounds.
- **Lack of Machine Learning Integration:** This method does not utilize machine learning or deep learning techniques, which are more accurate and robust in varied and dynamic conditions.
- **Sensitivity to Noise and Artifacts:** The approach is sensitive to noise and image artifacts, leading to potential false positives or missed detections.
- **Limited Generalization:** The method may not generalize well to different vehicle types, countries, or license plate formats.

In conclusion, both the traditional edge detection-based approach and the deep learning-based method offer distinct advantages for license plate detection and recognition. The edge detection approach, while simpler and faster, relies heavily on the quality of the input image and is prone to errors in challenging conditions such as poor lighting, motion blur, or complex backgrounds. On the other hand, the deep learning-based method, particularly with models like YOLOv8 for detection and PaddleOCR for recognition, demonstrates higher accuracy and robustness in varied conditions. It excels in detecting plates even in noisy environments or complex scenarios, thanks to its ability to learn from large datasets and adapt to different patterns. While the deep learning method requires more computational resources and training data, it ultimately provides superior precision, recall, and overall performance. Therefore, while the traditional method may be suitable for simpler, controlled environments, the deep learning-based approach is more effective in real-world applications where variability and complexity are greater.

## References

[1] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[2] Joseph, J., Berthold, P. (2015). Image Processing and Computer Vision: Principles and Applications. Springer.

[3] Sun, S., Wang, Z., Wang, J. (2018). A comprehensive review of optical character recognition. In Journal of Computer Science and Technology, 33(6), 1060-1075.