UCCD3074 Deep Learning for Data Science
Group Assignment
Deep Learning-Based Car Plate Detect Recognition System

Research-based [ ] Application-based [/]

Name	Brandon Kong Chen Wu (L)	Chin Zi Wei	Ting Jun Jing	Look Zheng Hong
Programme	CS	CS	CS	CS
ID	2300303	2105873	2300322	2106397
Contribution	1/4	1/4	1/4	1/4

1. INTRODUCTION

Car detection and license plate recognition are essential components of intelligent transportation systems, supporting applications such as vehicle access control, smart parking, enforcement of traffic rules, and automated tolling. Fast and precise vehicle identification not only reduces the dependence on manual monitoring but also enhances the scalability and reliability especially during heavy traffic.

Problem Statement

Car-plate recognition is a challenging task due to the need for both accurate detection and reliable character recognition under real-world conditions. Variations in plate formats, fonts, and sizes across regions, as well as environmental factors such as illumination changes, weather conditions, and motion blur, reduce system accuracy. Achieving high performance while maintaining real-time processing capability is the central problem this project seeks to address [1].

Motivation

Manual vehicle identification is labor-intensive, error-prone, and impractical for large-scale deployment. Automated License Plate Recognition (ALPR) systems offer significant benefits, including reduced operational costs, all time monitoring, and faster decision-making in traffic management. The growing demand for smart city infrastructure, intelligent parking, and security surveillance further highlights the importance of robust ALPR solutions. By leveraging open-source deep learning models such as YOLOv8 for license plate detection and PaddleOCR for character recognition, this project aims to build an affordable, accurate, and scalable pipeline for automated license plate recognition [2].

Background

Traditional approaches employed for the detection of plates included edge detection, color segmentation, and morphological operations. Although these methods were effective in a controlled environment, they became significantly less precise in real environments. But with the advent of maturity of deep learning, object detection algorithms such as Faster R-CNN, SSD, and YOLO have gained tremendous robustness and speed. YOLO (You Only Look Once) has been a favorite because it can be run in real time on images. However, detection alone is insufficient. Optical Character Recognition (OCR) also needs to be applied in order to read

alphanumeric text out of plates that have been detected. Combining YOLO for detection with OCR packages such as PaddleOCR provides a complete end-to-end solution [3].

Challenges

The main challenges in license plate recognition include:

- Varying illumination conditions and weather effects
- Different plate sizes, fonts, and formats across regions
- Motion blur in video sequences
- Occlusion and skewed viewing angles
- Real-time processing requirements for video applications
- False positives from text-like objects in the scene

Scope and Objectives

This project focuses on building a comprehensive license plate detection and recognition system with the following objectives:

- Detect license plates in both static images and video streams
- Extract text from detected plates using OCR engines
- Track vehicles across video frames using object tracking algorithms
- Evaluate system performance using standard metrics
- Handle real-world challenges such as varying lighting conditions and plate orientations

2. RELATED WORK

Salsabila and Sriani [4] presented a system for automatic license plate recognition (ALPR) that incorporates YOLOv8 for object detection and EasyOCR for recognition in handling 4K video inputs. Their work showed that YOLOv8 had the capabilities for effective and real-time license plate localization and hence the value in coupling lightweight deep learning detection with off-the-shelf OCR. The design of the pipeline was kept very simple and intentional and included plate localization, image pre-processing, and text reading, the sum total of which helped make the system very effective and replicable.

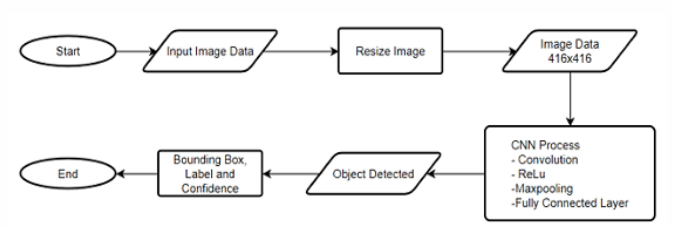


Figure 2.1: YOLO algorithm flowchart illustrating the detection process [4]

The primary advantages of their system are rooted in its straightforwardness and effectiveness. Through the utilization of YOLOv8, the framework demonstrated robust detection capabilities while maintaining low computational demands. However, as it relies on EasyOCR as a generic recognition component, the system lacked domain-specific optimization and sometimes struggled with non-standard or degraded license plates,

Despite these benefits, the system underperformed in the OCR stage. Recognition accuracy decreased in challenging situations such as motion blur, low illumination, and the occurrence of minor or sloping license plates. The approach also lacked effective mechanisms in the form of powerful preprocessing techniques, different OCR engines, or strategies for dealing with temporal consistency and gave varying outputs while processing video streams. The evaluation

was also limited to studies in feasibility and visual examination, without thorough quantitative measures of the accuracy of recognition.

In contrast, the present project enhances the baseline by training and evaluating multiple YOLOv8 variants (Models A–C), integrating PaddleOCR as the dedicated text recognition engine to achieve higher robustness on diverse license plate styles, and applying preprocessing methods such as CLAHE and adaptive padding to improve robustness. For video input, SORT tracking and OCR caching are employed to reduce redundant recognition and stabilize output. Unlike the earlier work, this project also introduces a detailed evaluation framework including exact match rates, character-level accuracy, and Levenshtein edit distance, as well as incorporates practical deployment features such as data augmentation, adjustable input sizes, and annotated video outputs. These improvements increase system complexity but provide stronger robustness, stability, and empirical validation compared to existing approaches.

3. SYSTEM DESIGN

Dataset & Preparation

We trained and evaluated the system on a custom dataset of vehicle images downloaded from Roboflow and four videos capturing car plates.

For images, each image in the dataset has one or more cars visible, and the license plate in each car is marked with a bounding box. The dataset was split into training, validation and test sets. The training set was used to fine-tune the YOLO detector, which the test set was reserved for evaluating the end-to-end system performance. In total, the test set there were around 20 unique license plates to recognize. Annotation file in YOLO format provided ground truth bounding boxes for plates and a separate ground truth CSV listed the correct plate string for each test image. Before training, images resized to a consistent dimension expected by the model.

System Overview

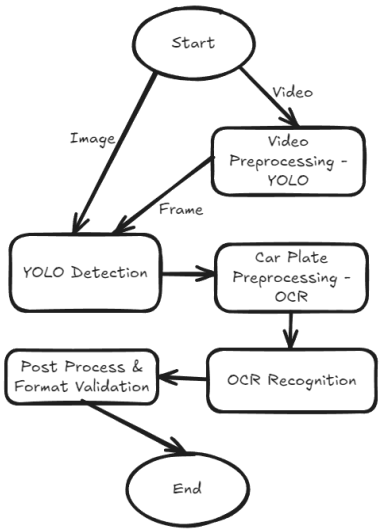


Figure 3.1: System Overview

Our license plate recognition system follows a two stage deep learning pipeline. First, a plate detection stage uses a YOLOv8 model to localize license plates in the input image. Second, a character recognition stage uses an OCR model (PaddleOCR) to read the text from each cropped plate region. The system was implemented in Python using the Ultralytics YOLOv8 library for detection and PaddleOCR library for text recognition. The overall design is modular.

The detector takes an image and produces a bounding box coordinates for any license plates. Then, the OCR module processes those cropped regions to output text strings. This modular approach allows each component to be improved or replaced independently.

Video Preprocessing

While detection models are effective on individual frames, running them independently on each frame introduces challenges. The same car plates may be detected multiple times, leading to duplicate redundant recognition. To address these challenges, we introduced a tracking mechanism that maintains consistency across frames. Instead of treating each frame as independent, the tracker links detections over time and assigns a stable identity to each vehicle.

This provides several important benefits:

- Reduced redundant OCR calls: By remembering the ID and caching the text, the system does not need to perform OCR on the same plate in every single frame. Instead, once a plate is recognized with sufficient confidence, the cached result can be reused. Only when the confidence is low due to some condition, the OCR will perform again in the next frame to gain a more accurate prediction
- Better handling of motion: Even when vehicles move quickly, or detection might briefly fail for a frame, the tracker helps maintain continuity by reassigning the same ID once the object reappears.

YOLO Detection Module

The detection stage of the system is responsible for locating vehicle number plates within each image or video frame. To accomplish this, we employ the YOLO carplates detection model, which is well suited for real-time applications because it performs detection in a single pass through the network.

When an image or frame is processed, YOLO outputs a collection of bounding boxes**,** rectangular regions that indicate where plates are likely to be present. Each bounding box is associated with a confidence score that reflects how certain the model is about the detection. To maintain accuracy, we apply a confidence threshold, ensuring that only predictions above a certain probability are considered valid. The system simply skips OCR for non-detected frame, which prevents unnecessary processing and false results.

By filtering out low-confidence detections and redundant boxes, the system ensures that the subsequent OCR stage receives clean and focused input, improving both recognition speed and accuracy.

Preprocessing for OCR

The first step of preprocessing is **"**Region of Interest Extraction**"**. Because small plates are especially vulnerable to OCR errors when tightly cropped, the implementation introduces content-aware padding. If the detected box is smaller than 150×100 pixels, the crop is expanded by 20% on all sides, clipped to image boundaries. Otherwise, the exact detected box is used. This design helps to preserve character ascenders, descenders and mitigate minor localization errors, both of which are known to harm text recognition when crops are overly tight.

Next, each plate crop is converted to grayscale and contrast-enhanced using CLAHE for improving local contrast under challenging illumination such as glare, shadows or low light.

Character Recognition

For reading text on detected plates, we integrate the PaddleOCR toolkit, a state-of-the-art open-source OCR toolkit that provides pre-trained models optimized for multiple languages. In our case, the English alphanumeric model was selected, as it is well suited to license plate formats that primarily consist of uppercase letters and digits.

The recognizer used is based on the Convolutional Recurrent Neural Network (CRNN) architecture. This design is advantageous for license plates because it combines convolutional layers, which extract spatial features of characters, with recurrent layers that capture sequential dependencies across the string. This makes the model robust against variations in plate fonts, spacing, and alignment.

Each car plate is submitted to the OCR engine, which returns a predicted string and an associated confidence score. Robustness checks ensure that empty or malformed OCR outputs are handled without raising errors.

Post-processing & Format Validation
First, trivial tokens that never appear in plate numbers (spaces, dots) are removed. Second, the code applies heuristic character substitutions to address common glyph ambiguities. For example, since 'I' and 'O' is not allowed in Malaysia car plate, thus 'I' and lowercase 'l' are mapped to 1, and 'O' is mapped to 'Q' in this implementation. Finally, the cleaned string is validated against a regular expression that encodes a Malaysian-style plate format, "^[A-Z]{1,3}\d{1,4}[A-Z]{0,2}$". By combining string cleaning, heuristic corrections, and strict format validation, the post-processing stage transforms noisy OCR predictions into reliable plate numbers while rejecting impossible results. This fusion of machine learning and rule-based domain knowledge ensures higher overall accuracy and trustworthiness of the recognition system.

4. EXPERIMENT & EVALUTION

Across all experiments, the evaluation considered two levels of performance. The detection quality of the license-plate localizer and the end-to-end quality after OCR and post-processing. Detection was assessed qualitatively by the presence of missed plates, spurious boxes, and bounding-box tightness, while end-to-end performance was analyzed using exact-match rate, character-level accuracy, and the average edit distance between predicted and true strings.

YOLO Detection Experiment Setup

We conducted multiple runs with varying freeze settings:

Model A:

To address the limited size of our dataset, we adopted a transfer learning strategy by freezing the majority of the YOLOv8 backbone during training. The backbone layers are primarily responsible for extracting generic visual features such as edges, textures, and shapes, which are common across most computer vision tasks. Since these features are already well-learned from the large-scale COCO dataset used in pre-training, retraining them on our smaller dataset would improve stability, as the model retained robust pre-trained features while adapting to the new domain. The trade-off of this method is that the model has less flexibility to adapt to dataset-specific challenges, such as very small or low-contrast plates. Therefore, it can miss or reject some valid plates.

Parameter	Value

Starting weight	yolov8n.pt
Image Size	640
Epochs	50
Frozen layer	10
Mosaic probability	1.0
HSV	0.015, 0.7, 0.4

Model B:

To overcome the trade-off of model A, we initially froze the backbone layers for the first 15 epoch with same setting as model A. Then we then unfreeze the backbone layers for the next 35 layers. This progressive unfreezing strategy combines the benefits of transfer learning (leveraging strong pre-trained features) with the flexibility of fine-tuning (adapting to dataset-specific nuances). It generally leads to more robust convergence, reduces the risk of catastrophic forgetting, and often results in higher accuracy compared to either fully freezing or fully unfreezing from the start.

Parameter	**Value**
Starting weight	(best weights of A)
Image Size	640
Epochs	35
Frozen layer	0
Mosaic probability	1.0
HSV	0.015, 0.7, 0.4

Model C:

Another model with full fine-tuning of YOLOv8n (no freezing) from the start was trained for comparison. We trained YOLOv8n for 50 epochs on the dataset, with freeze=0, which the entire model's weights were updated during training. The rationale behind this is that while freezing the backbone stabilizes early training, it also limits the network's ability to adapt its low-level features (edges, textures, small details) to the unique patterns in our dataset.

License plate detection has characteristics that may differ significantly from the generic objects in the COCO dataset that YOLO was pretrained on. For example, plates often contain high-contrast alphanumeric characters, small rectangular regions, and repetitive patterns. These are not strongly represented in COCO's categories. Full fine-tuning allows the entire model, including the early convolutional layers to adapt to these specific features, potentially improving recall on challenging cases like small, low-contrast, or partially occluded plates.

Parameter	**Value**
Starting weight	yolov8n.pt
Image Size	640
Epochs	50
Frozen layer	0
Mosaic probability	1.0
HSV	0.015, 0.7, 0.4

Effect of Preprocessing on OCR

To assess the contribution of preprocessing, we compared OCR results with and without padding, grayscale conversion, and contrast enhancement.

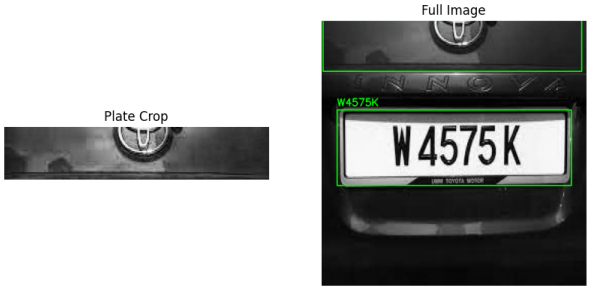
Overall Evaluation Results:

Model A	Pairs evaluated	21
	Exact Match Accuracy	0.62
	Character-Level Accuracy	0.67
	Avg. Edit Distance	3.46
Model B	Pairs evaluated	23
	Exact Match Accuracy	0.65
	Character-Level Accuracy	0.72
	Avg. Edit Distance	3.09
Model C	Pairs evaluated	24
	Exact Match Accuracy	0.58
	Character-Level Accuracy	0.67
	Avg. Edit Distance	3.12

YOLO Detection Evaluation

With only 21 plate pairs evaluated on model A which is lower compared to model B and model C. It shows the reliance on frozen pre-trained features limited adaptability, leading to more missed detections. Example of car plates that missed detections while other two models captured.

While model C is less robust to noisy, due to limited ability to adapt its low-level features, which leads to false detections. Example of false detections.

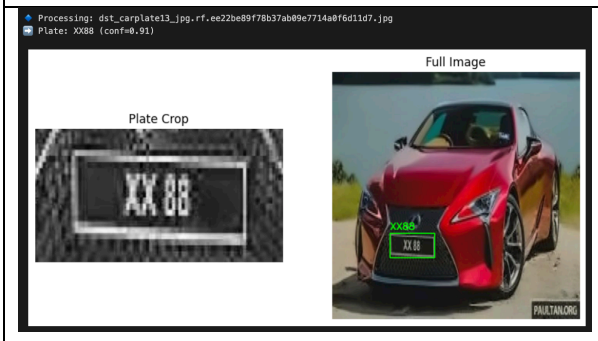
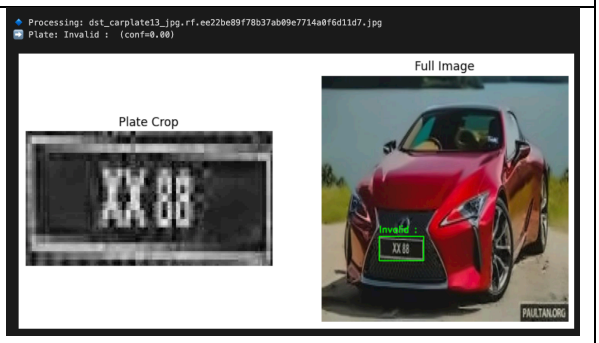


Comparatively, model B achieves the best balance. With progressive unfreezing, it leverages pre-trained features early while later adapting to dataset-specific challenges. Its higher scores (0.65 exact-match, 0.72 character-level) validate its robustness.

OCR Recognition Evaluation

	Model B	Model B (without padding)
Pairs evaluated	23	23
Exact Match Accuracy	0.65	0.61
Character-Level Accuracy	0.72	0.71
Avg. Edit Distance	3.09	3.09

Based on the results comparing Model B with and without padding, exact-match and character-level accuracy decreased to 0.61 and 0.71 respectively when tested without padding. This shows that padding adds a safety margin, so the OCR sees the complete glyphs, reduces aliasing.

Model B	Model B (without padding)
	
	

Real Time Video Preprocessing – Example

Summary

Model B (progressive unfreezing) gave the best balance, it leverages pretrained features early while later adapting to dataset-specific challenges. OCR results confirmed that preprocessing significantly improves recognition. Which are padding, grayscale conversion, and contrast enhancement. Overall, Model B with preprocessing was the most reliable configuration.

5. CONCLUSION

This project successfully developed an Automatic License Plate Recognition (ALPR) system by combining deep learning-based object detection and optical character recognition algorithms. A YOLOv8 model was trained and fine-tuned using a custom dataset to detect car license plates with high accuracy. PaddleOCR was used to extract alphanumeric characters from the detected regions, allowing the system to perform accurate plate number recognition in a variety of resolutions and lighting conditions.

The system was extended beyond static image analysis to include real-time video processing. A tracking technique enabled the system to continually watch vehicles across video frames while reducing inefficient OCR processes. This enabled the efficient recognition and tracking of several cars in dynamic situations, confirming the model's potential for real-world surveillance or traffic monitoring applications.

Overall, this project demonstrates the possibility of integrating recent deep learning object identification frameworks with OCR in intelligent transportation systems. The major findings show that YOLOv8 delivers accurate and efficient plate detection, whereas OCR approaches can consistently extract textual information in most instances.

REFERENCES

[1] U. Saha, A. Pramanik, A. K. Maji, and A. Mukherjee, “A Practical Weather-Adaptive CNN for License Plate Detection,” *Journal of Imaging*, vol. 10, no. 7, p. 129, 2024. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11644901

[2] P. Batra, D. Choudhary, S. Verma, and S. S. Chouhan, “A Novel Memory and Time-Efficient ALPR System Using YOLOv5,” *Journal of Imaging*, vol. 8, no. 7, p. 191, 2022. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC9317241

[3] H. Moussaoui, M. A. Reda, and M. A. Mohamed, “Enhancing Automated Vehicle Identification with a YOLO-v8 Method, Image Processing, and OCR,” *Scientific Reports*, vol. 14, no. 1, pp. 1–14, 2024. [Online]. Available: https://www.nature.com/articles/s41598-024-65272-1

[4] N. Salsabila and S. Sriani, “Enhancing Automated Vehicle License Plate Recognition with YOLOv8 and EasyOCR,” Journal of Information Systems and Informatics, vol. 6, no. 3, pp. 741–752, Sept. 2024.