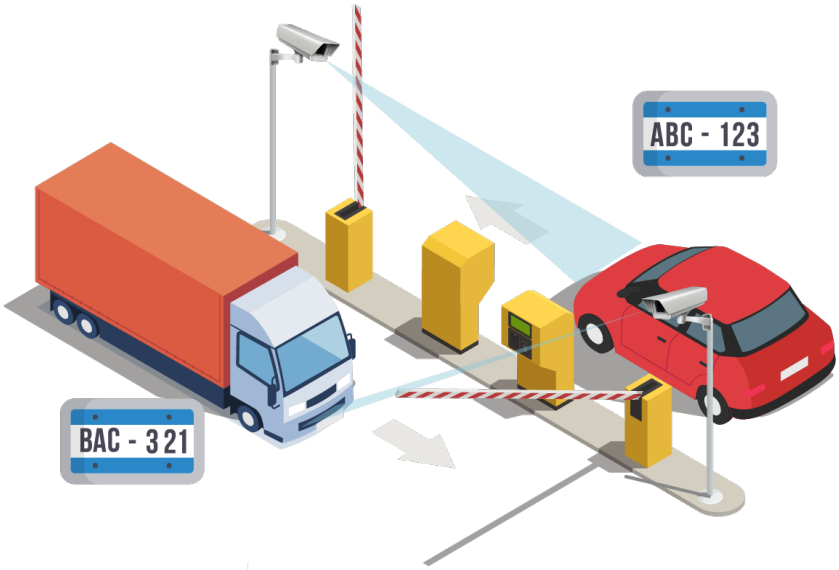
ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA



Development of License Plate Detection and Recognition System based on Deep Learning

BRUNO TIAGO CAMPOS COLAÇO

(Licenciado)

Trabalho de Projeto para obtenção do Grau de Mestre em

Engenharia Informática e Multimédia

Orientadores: Doutor Gonçalo Caetano Marques
Doutor Pedro Miguel Torres Mendes Jorge
Jeremy Kespite

Júri:

Presidente: Doutor Carlos Jorge de Sousa Gonçalves
Vogais: Doutor Arnaldo Joaquim Castro Abrantes
Doutor Pedro Miguel Torres Mendes Jorge

Setembro 2024

Development of License Plate Detection and Recognition System based on Deep Learning

BRUNO TIAGO CAMPOS COLAÇO

(Licenciado)

Trabalho de Projeto para obtenção do Grau de Mestre em
Engenharia Informática e Multimédia

Orientadores: Doutor Gonçalo Caetano Marques, ISEL
Doutor Pedro Miguel Torres Mendes Jorge, ISEL
Jeremy Kespite, Europol

Júri:

Presidente: Doutor Carlos Jorge de Sousa Gonçalves , ISEL
Vogais: Doutor Arnaldo Joaquim Castro Abrantes, ISEL
Doutor Pedro Miguel Torres Mendes Jorge, ISEL

Setembro 2024

Abstract

Automatic License Plate Recognition (ALPR) systems play a vital role in various applications, including traffic management, law enforcement, and parking control. This thesis presents a comprehensive study on the development of a robust ALPR system based on deep learning techniques. The proposed system is structured in three main stages: (1) vehicle detection and classification, where vehicles are identified and categorized into cars, buses, trucks, and motorcycles using three convolutional neural network (CNN) models—Faster R-CNN, YOLOv8, and SSD; (2) license plate detection, achieved through a pretrained YOLOv8 model, which isolates license plates from detected vehicles; and (3) character recognition, utilizing Tesseract OCR to extract alphanumeric characters from the detected license plates. The project emphasizes the importance of integrating high-accuracy detection models to enhance the overall performance of the system. Performance metrics, including Character Error Rate (CER), mean Average Precision (mAP), and Intersection over Union (IoU), are employed to evaluate the effectiveness of each stage. This research ultimately aims to contribute to the existing body of knowledge in ALPR systems by proposing an efficient, accurate, and scalable solution for real-world applications.

Keywords: Deep Learning, Vehicle Detection, License Plate Detection, OCR, Automated License Plate Recognition (ALPR)...

Resumo

Os Sistemas de Reconhecimento Automático de Placas (ALPR) desempenham um papel vital em várias aplicações, incluindo gestão de tráfego, segurança pública e controle de estacionamento. Esta tese apresenta um estudo sobre o desenvolvimento de um sistema robusto de ALPR baseado em técnicas de aprendizagem profunda. O sistema proposto é estruturado em três etapas principais: (1) detecção e classificação de veículos, onde os veículos são identificados e categorizados em carros, autocarro, camiões e motos utilizando três modelos de rede neural convolucional (CNN)—Faster R-CNN, YOLOv8 e SSD; (2) deteção de placas, realizada através de um modelo YOLOv8 pré-treinado, que isola as placas detectadas dos veículos; e (3) reconhecimento de caracteres, utilizando Tesseract OCR para extrair caracteres alfanuméricos das placas detectadas. O projeto realça a importância de integrar modelos de detecção de alta precisão para melhorar o desempenho geral do sistema. As métricas de desempenho, incluindo Taxa de Erro de Caracteres (CER), mean Average Precision (mAP) e Interseção sobre União (IoU), são utilizadas para avaliar a eficácia de cada etapa. Esta pesquisa visa, em última análise, contribuir para o corpo de conhecimento existente em sistemas de ALPR, propondo uma solução eficiente, precisa e escalável para aplicações do mundo real.

Palavras-chave: Aprendizagem Profunda, Deteçãode de Veículos, Deteçãode de Matrículas, Reconhecimento Ótico de Caracteres, ALPR ...

Contents

List of Figures

List of Tables

1 Introduction

The increasing number of vehicles on the road has led to a growing demand for automated systems that enhance traffic management, security, and law enforcement operations. One such system is Automated License Plate Recognition (ALPR), which plays a crucial role in applications such as toll collection, parking management, traffic monitoring, and criminal investigations. Traditional ALPR systems, often relying on manual processes or outdated image processing techniques, face limitations when applied to real-world conditions like poor lighting, angled views, or varying license plate designs.

Recognizing the need for advanced and scalable ALPR systems, Europol has proposed the development of systems capable of monitoring vehicles. However, most existing ALPR systems are built using private datasets, which restricts their adaptability and are hard to reproduce. This thesis aims to address these limitations by developing an open-source License Plate Detection and Recognition System based on deep learning techniques, relying entirely on publicly available datasets and tools.

The system utilizes convolutional neural networks (CNN) for vehicle and license plate detection, coupled with optical character recognition (OCR) for accurate character recognition. Designed to handle diverse real-world conditions, the system is trained on a large, varied dataset that captures different lighting, angles, and license plate formats. By using open-source resources, this project ensures that the system can be accessed and improved upon by the broader research community.

Another contribution of this project is the development of a complete dataset that covers the entire pipeline, from vehicle detection and classification to character recognition. This dataset not only ensures the robustness of the system but also provides a valuable resource for future research in the field of intelligent transportation systems and computer vision.

The document is structured as follows: Chapter 2 reviews related work in the field of license plate recognition, focusing on both single-stage and multi-stage approaches.

Chapter 3 describes the methods used in this project, including the system architecture, vehicle detection, license plate detection, and character recognition stages. Chapter 4 presents the experimental results and evaluates the system's performance using key metrics. Finally, Chapter 5 discusses conclusions and potential future work, including further improvements to the system and the exploration of new applications.

2 Related Work

This chapter reviews the latest and older techniques for ALPR. It will review approaches that use traditional image processing, machine learning, and deep learning. In particular, it will highlight recent advances using CNNs and how they improve license plate detection and character recognition. ALPR systems are divided into two categories: multi-stage and single-stage. A single-stage ALPR system handles the entire process of license plate recognition—detecting the vehicle, identifying the plate, and reading the characters—all in one step. A multi-stage ALPR system breaks the process into separate steps: first detecting the vehicle, then locating the license plate, and finally recognizing the characters on the plate.

This chapter aims to provide a clear understanding of current trends and challenges in the field of ALPR as well as the evolution.

2.1 Multi-stage

Most of the ALPR systems use a multi-stage approach each consists on three main stages: license plate detection, character segmentation and character recognition.

2.1.1 License Plate Detection

License plate detection in multi-stage systems has evolved through various techniques over time, each addressing specific challenges such as noise, lighting conditions, and varying plate designs.

2.1.1.1 Edge-Based Methods

One of the foundational approaches for license plate detection involved edge-based methods, which takes advantage of the rectangular shape, and the boundary of the license plate compared to the vehicle body is very different. Horizontal edges or vertical edges can be detected. Several studies have used the Sobel filter to detect the edges of a license plate [1] [2] [3] [4] utilizing the aspect ratio of the license plate

to identify candidate regions. The Sobel filter consists of two 3 x 3 convolutional matrices (one for vertical edges and another for horizontal edges). This approach, although easy to use, is very sensitive to noise. According to [3], this method had a success rate of 96.2%.

Another method developed in [5] consists of a combination of a line grouping algorithm and an edge density mapping algorithm. The line grouping algorithm identifies and clusters line segments along the license plate boundary, while the edge-density map detects areas with a high concentration of lines as potential license plate regions. This new algorithm has shown a success rate of 99.45%.

The Hough transformation can be used to detect straight lines in the image, but this method is sensitive to deformations in the boundary. In [5] a method is proposed in which the Hough transform is combined with a contouring algorithm that achieved an accuracy of 98.8%.

2.1.1.2 Color-Based Methods

Color-based detection leverages the distinct color contrast between the license plate and the vehicle body. This approach is particularly useful in scenarios where the license plate color is standardized.

In [6] is proposed an algorithm called Gaussian Weighted Histogram Intersection (GWHI) for classifying license plates. The traditional histogram intersection technique, which compares two colors by matching their color histograms, is typically sensitive to changes in illumination. To address this issue, a Gaussian function is used, thereby improving its performance under varying lighting conditions.

Another license plate localization technique using mean shift segmentation is discussed in [7]. In this approach, vehicle images are segmented into regions of interest using the Mean Shift algorithm, which relies on the color information in the image. These segmented regions are then classified with a Mahalanobis classifier to accurately detect license plates.

Color-based methods can be effective for detecting deformed or inclined license plates, but they are rarely used alone due to their sensitivity to changes in light. Additionally, their performance is influenced by the specifications of the camera used to capture the images. These methods can also produce errors if the image contains other regions with colors similar to the license plate. As a result, they are often combined with other techniques to ensure accurate detection.

2.1.1.3 Texture-Based Methods

Texture-based methods focus on the unique textural features of license plates, which differ from the surrounding vehicle body due to the presence of characters. Because

of the contrast between the plate and its characters, it creates a unique pixel intensity distribution around the plate.

One technique often used in texture analysis is the Gabor filter. Gabor filters are linear filters used for texture analysis, which can capture various frequencies and orientations present in the image. They are particularly effective in detecting the repetitive patterns and textures associated with the characters on a license plate. It is applied to the image in multiple orientations and scales to extract features that correspond to the texture of the license plate. The filter is essentially a sinusoidal wave modulated by a Gaussian function, making it sensitive to specific frequency and orientation information. The use of the Gabor filter is described in [8]. Although this method is robust to variations in lighting and color because it relies on the frequency domain rather than direct pixel values, it is computationally intensive and may produce false positives if other regions of the image have similar textural patterns.

In [9] is proposed the use of Wavelet Transform. Wavelet Transform is a mathematical tool that decomposes an image into different frequency components, allowing the analysis of textures at various levels of detail. In license plate detection, wavelet transform is used to identify regions that exhibit the high-frequency content characteristic of license plate textures. The image is decomposed into sub-bands, typically into four categories: Low-Low (LL), High-Low (HL), Low-High (LH), and High-High (HH). The HL and LH sub-bands are particularly useful for detecting the horizontal and vertical edges of the characters on the plate. The reported detection accuracy is 92,4%. This method provides multi-resolution analysis, allowing for the detection of textures at various scales, which is useful for detecting license plates at different distances. However it is sensitive to noise especially in images with complex backgrounds.

In [10] a segmentation technique is used Sliding Concentric Windows for license plate detection. This method achieved properly segmented 96,5% of the images.

2.1.1.4 Character-Based Methods

Character-based methods operate on the premise that the most distinguishing feature of a license plate is the sequence of alphanumeric characters it contains. These methods focus on identifying regions in the image that exhibit properties typical of text, such as consistent patterns, spacing, and contrast relative to the background. Once these regions are identified, further analysis is conducted to verify that they indeed contain valid license plate characters.

Matas and Zimmermann [11] proposed a method that extracts all character-like regions in an image. These regions are then classified using a neural network, and

those forming a linear spatial configuration are assumed to be part of a license plate. This approach is robust against different illumination conditions and viewpoints, achieving high detection accuracy. This method achieved 95% accuracy.

In [12], Draghici implemented a horizontal scanning method to identify repeating contrast changes with a minimum length of 15 pixels. This method operates on three primary assumptions: (1) there is adequate contrast between the background and the characters, (2) the license plate contains at least 3-4 characters, and (3) the minimum vertical size of a character is 15 pixels. However, the required minimum vertical size may vary depending on the camera specifications and needs to be re-calibrated with any changes in hardware. This approach reportedly achieves a 99% detection rate in outdoor environments.

2.1.1.5 Statistical Classifiers

One approach is to use Haar-like features with Adaptive Boosting (AdaBoost) to train cascade classifiers for license for license plate detection. A decision tree based cascade classifier with AdaBoost training is proposed in [13]. To train, a Haar-like features and statistical features are used. This algorithm has a detection rate of 94.5%.

Kim et al. [14] uses an algorithm based on color texture for license plate detection. A Support Vector Machine (SVM) is used to classify a region as license plate or not using its color and texture properties. After the SVM each pixel indicates the probability for it being a part of the plate and with that is possible to predict the bounding box.

2.1.1.6 Deep Learning Techniques

Deep learning has revolutionized the field of Automated License Plate Recognition ALPR, especially in the detection phase, by enabling more accurate, robust, and scalable solutions compared to traditional methods. Deep learning-based approaches leverage large datasets and advanced neural network architectures to automatically learn the complex features necessary for detecting license plates under various conditions.

CNN can be used to detect the location of the plate, like studied in [15]. In this study the detection was divided into two steps. The first one is to pre-process the input image to remove noise and to extract details from the image. In the second step candidate regions are extracted and classified as a plate region or not using a CNN classifier. This study got 93,8% of recall and 91,3% of f-score.

Another study that used CNN was done in [16]. In this study they used two CNNs, one shallow CNN and another deep CNN. The shallow CNN was used to remove

background regions from the image to reduce computational cost while the deep CNN is used to detect license plates from the remaining regions. The experimental results have shown above average results for accuracy with a less computational cost.

Several studies have used state-of-the-art YOLO object detector for license plate. In [17] they used a YOLO (version 2) object detector to build a system for plate detection. They use two separate CNN one to detect vehicles another to detect license plates. The CNN to detect license plates uses as input images the vehicles detected by the first CNN.

In [18], based on the original YOLO, they improved it to handle multi-directions. YOLO only gives information about the object coordinates (the center), height and width, but with this improved version it also gives the information of the angle of rotation.

2.1.2 Character Segmentation

Character segmentation is a critical stage in Automated License Plate Recognition ALPR systems, responsible for isolating individual characters from the license plate for subsequent recognition. Various techniques have been developed and refined over the years, addressing the challenges posed by diverse environmental conditions and varying license plate designs.

2.1.2.1 Pixel Connectivity

Pixel connectivity is a simple method used for segmenting characters. In [19], connected pixels are labeled, and those forming objects of predetermined size or aspect ratio are extracted as characters. This method is robust against rotated license plates but struggles with broken or merged characters due to binarization issues.

2.1.2.2 Projection Profiles

Projection profiles, particularly vertical and horizontal projections, have also been used to detect the starting and ending positions of characters on the license plate. This technique benefits from the contrasting colors of the characters and background after binarization. However, projection-based methods are sensitive to image noise and quality, often requiring a noise reduction stage in the pre-processing phase to enhance segmentation accuracy. For example, in [20] vertical projection was utilized to detect the starting and ending positions of the characters, while horizontal projection was to detect the characters. This method proved effective in distinguishing between the characters and the plate background in binary images.

2.1.2.3 Character Segmentation Using Prior Knowledge

Character segmentation techniques that leverage prior knowledge about the license plate, such as the aspect ratio of characters or specific color distributions, have also been employed. Busch et al. [3] for example, proposed a method that involved scanning the binary image horizontally to find regions where the pixel ratio of background to character pixels exceeded a threshold, thereby identifying the character boundaries.

Another approach by Paliy et al. [21] involved resizing the extracted license plate to a fixed size where the character positions were predetermined, simplifying the segmentation process. While these methods are relatively simple and effective, they tend to be region-specific and may not generalize well across different license plate types.

2.1.2.4 Deep Learning Approaches

With the rise of deep learning, character segmentation has seen substantial improvements in accuracy and robustness. (CNNs) have become a popular choice for character segmentation tasks in ALPR systems. These networks are trained to receive a localized license plate as input and produce bounding boxes for each character as output, effectively segmenting characters without the need for extensive pre-processing stages [17].

The success of this approach its going to depend on the dataset that the CNN was trained on.

2.1.3 Character Recognition

Character recognition is a crucial component of Automated License Plate Recognition ALPR systems, responsible for interpreting the segmented characters from a license plate into readable text. Over the years, a range of techniques has been employed for this task, from traditional pattern matching to advanced deep learning methods. This section reviews various approaches, referencing specific studies that have implemented these methods.

2.1.3.1 Template and Pattern Matching

Early approaches to character recognition often relied on template matching techniques due to their simplicity and effectiveness in scenarios where the font and character size of license plates are consistent. In template matching, each character extracted from the license plate is compared against predefined templates representing possible characters. Metrics like Mahalanobis distance, Jaccard similarity,

and normalized cross-correlation are commonly used to determine the best match between a character and the templates.

For example, in [4] utilized template matching by comparing the segmented characters with stored templates, achieving reliable recognition for standardized fonts used in specific regions. However, these methods are limited by their inability to generalize across different plate designs and are computationally expensive when dealing with a large number of templates or when handling rotated characters.

2.1.3.2 Feature Extraction and Machine Learning

As ALPR systems evolved, more sophisticated approaches began to emerge, particularly those involving feature extraction combined with machine learning techniques. Feature extraction techniques reduce the dimension of the input data by identifying key characteristics of the characters, such as edges, shapes, and textures, which are then used for classification.

Gabor filters [22], Kirsh edge detectors [23] and eignvector transformation [24] are examples of feature extraction techniques that have been used to enhance character recognition. These methods are particularly useful in handling variations in character appearance caused by noise, rotation, or distortion. These extracted features are then classified using machine learning models, such as SVM.

2.1.3.3 Deep Learning

The advent of deep learning has revolutionized character recognition in ALPR systems. CNNs, in particular, have demonstrated exceptional performance by automatically learning to extract features directly from the raw pixel data, thus eliminating the need for manual feature extraction.

For instance, in [25] applied a CNN-based approach for character recognition in license plates, achieving state-of-the-art accuracy. The CNN model was trained to recognize characters in a variety of challenging conditions, including different lighting, occlusions, and distortions, making it robust and highly adaptable.

2.2 Single-stage

Single-stage ALPR systems represent a significant advancement in the field, streamlining the traditional multi-stage processes into a single, end-to-end framework. These systems perform license plate detection, localization, and character recognition in one integrated step, making them faster and often more efficient compared to their multi-stage counterparts.

Single-stage ALPR systems simplify the pipeline by integrating all necessary processes—detection, segmentation, and recognition into a single neural network. This

approach leverages the strong correlation between license plate detection and character recognition, allowing for shared parameters and reduced model complexity.

2.2.1 Early Developments

The earliest attempts to develop single-stage ALPR systems were driven by the need for more efficient processing, especially for real-time applications. Li et al. [26] were among the first to propose a single-stage ALPR approach. They utilized a modified VGG16 network for feature extraction, reducing the number of pooling layers to better capture the small and dense regions typical of license plates. The system employed a Region Proposal Network (RPN) to generate candidate regions for license plates, which were then processed for both detection and recognition in a single forward pass.

This pioneering work demonstrated that single-stage methods could achieve high accuracy while being computationally efficient, laying the foundation for subsequent research in this area.

2.2.2 Advances with CNNs

Building on the success of earlier models, subsequent studies have enhanced single-stage ALPR systems by refining their neural network architectures. For instance, Xu et al. [27] proposed a simplified convolutional neural network CNN with only 10 layers, specifically designed to predict bounding boxes directly. Their approach incorporated outputs from multiple layers of the CNN, each handling different receptive field sizes, enabling the detection of license plates at varying distances.

This model was notable for its simplicity and speed, as it avoided the use of complex recurrent networks, relying instead on individual classifiers for each character in the license plate. This approach significantly reduced processing time and made the model suitable for deployment in real-time applications.

2.2.3 Modern Approaches

More recent developments in single-stage ALPR have focused on deep learning models that further integrate the detection and recognition stages. Zhou et al. [16] introduced an end-to-end system based on You Only Look Once (YOLO) architecture, which processes the entire image in a single pass through the network. YOLO-based models, known for their speed and accuracy in object detection, have been adapted to handle the specific requirements of ALPR, such as recognizing small and densely packed characters.

2.3 Datasets

The development of robust ALPR systems has been significantly enhanced by the availability of large-scale datasets. In this section, it is going to discuss how some of the most popular available datasets are used to train ALPR systems.

2.3.1 COCO Dataset

The COCO (Common Objects in Context) [28] dataset is a large-scale object detection, segmentation, and captioning dataset widely used for training deep learning models in computer vision tasks. It contains over 330,000 images, out of which more than 200,000 are labeled with more than 1.5 million object instances across 80 object categories. COCO is designed to reflect real-world scenes and contains objects in various challenging contexts, such as partial occlusions, cluttered environments, and objects of varying sizes and scales.

The COCO dataset has been widely utilized for vehicle detection in numerous studies due to its rich collection of labeled images across various object categories, including vehicles. In [28] the authors used the COCO dataset to test their vehicle detection model, showcasing its performance in unconstrained environments. This study emphasized the COCO dataset's suitability for training models to detect vehicles in real-world scenarios.

2.3.2 OpenImages Dataset

The OpenImages dataset [29] is a large-scale collection of annotated images designed for object detection, segmentation, and visual relationship detection tasks. Developed by Google, it contains millions of images with rich annotations, including object bounding boxes, labels, and instance segmentation masks. OpenImages stands out for its extensive range of objects, covering a diverse set of categories such as vehicles, animals, and everyday objects. It is especially useful for vehicle detection and license plate recognition because, not only has classes like car, bus, truck and motorcycle, it also has a license plate class.

This paper utilized the OpenImages V6 dataset for vehicle license plate detection, integrating YOLO as the object detector. The study highlights the effectiveness of the dataset in providing diverse vehicle images with accurate license plate annotations [29].

3 Methods

The development of the proposed License Plate Detection and Recognition System is structured into three main stages: vehicle detection and classification, license plate detection, and character recognition. Each stage leverages deep learning techniques to ensure robust detection and recognition under various conditions. The following sections provide a detailed explanation of each stage, the models used, and the tools implemented.

The code for implementing the license plate detection and recognition system is available in the following Github repository: License-Plate-Detection-and-Recognition-System-based-on-Deep-Learning.

3.1 System Architecture

The system is composed of three primary stages: *vehicle detection and classification*, *license plate detection*, and *character recognition*. Each stage is responsible for a specific part of the overall task and operates in sequence to achieve accurate license plate recognition.

In this project, a multi-stage approach was adopted for the ALPR system to ensure scalability and modularity. This design allows each stage—vehicle detection, license plate detection, and character recognition—to function independently. As a result, individual components can be upgraded or replaced without impacting the performance or structure of other stages in the system. This modularity enhances the system's adaptability and future development potential.

The overall workflow of the system can be summarized as follows:

1. An input image is fed into the system, which passes it to the vehicle detection and classification stage.

2. Detected vehicles are classified and their corresponding bounding boxes are

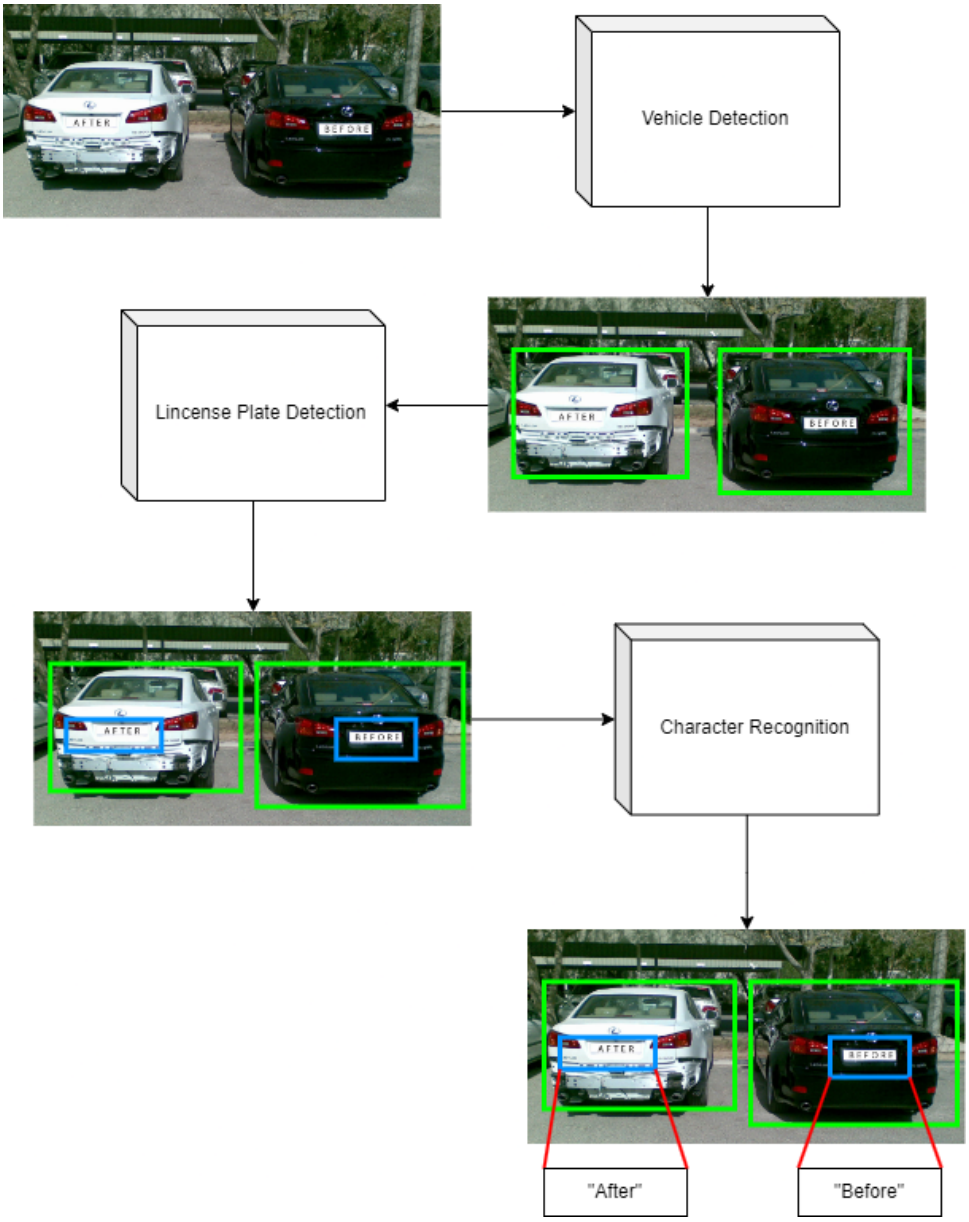


Figure 3.1: *Diagram of the system architecture.*

used to crop vehicle regions from the original image.

3. Cropped vehicle images are then passed to the license plate detection stage, where bounding boxes for the license plates are predicted.

4. The regions containing the license plates are cropped, and these cropped images are passed to the character recognition stage.

5. Character recognition is performed on the cropped license plate images, resulting in the extraction of the alphanumeric text.

A visual representation of this system flow is illustrated in Figure 3.1.

Below it is described what is the input, objective and the output of each stage.

3.1.1 Stage 1: Vehicle Detection and Classification

The first stage in the system pipeline is responsible for detecting and classifying vehicles within the input image. Given a raw input image, the goal is to locate all vehicles present in the image and classify each one into one of four predefined categories: *car*, *bus*, *truck*, or *motorcycle*, this classes were chosen because it encompass all the main vehicles on the road. This stage is crucial because it isolates the regions of interest (vehicles) from the rest of the image, thereby reducing the area to search for license plates in the following stage.

- **Input:** The raw image.
- **Objective:** Detect and classify the vehicles in the image. Crop image according to the bounding boxes of the vehicles detected.
- **Output:** Individual images containing just the detected vehicles.

By detecting and cropping vehicles from the input image, the system effectively reduces the search space for the next stage, improving both the speed and accuracy of license plate detection.

3.1.2 Stage 2: License Plate Detection

The second stage processes the cropped vehicle images from the previous step to detect the license plates within those images. This step ensures that the region containing the license plate is accurately localized.

- **Input:** The cropped images of detected vehicles from the previous stage.
- **Objective:** Detect the license plates from the input images and crop the image according to the bounding boxes of the license plates.
- **Output:** Individual images containing just the detected license plates.

This step isolates the license plate region, ensuring that the subsequent character recognition stage can focus solely on extracting the plate's alphanumeric content.

3.1.3 Stage 3: Character Recognition

The final stage of the system focuses on recognizing the characters from the cropped license plate image. This stage is responsible for accurately extracting the alphanumeric text from the plate, which is the ultimate goal of the system.

- **Input:** Takes as input individual images containing the detected license plates.

- **Objective:** Extract the alphanumeric text from the license plate.
- **Output:** The alphanumeric text from the image.

3.2 Vehicle Detection and Classification

In the vehicle detection and classification stage, three different Convolutional Neural Network (CNN)-based models were tested: **Faster R-CNN**, **Single Shot Detector (SSD)**, and **YOLOv8**. These were selected for their established performance in object detection tasks and were all pretrained on the **COCO dataset**, which includes diverse categories, including vehicles that is relevant for this system.

3.2.1 Faster R-CNN

Faster R-CNN (Region-Convolutional Neural Network) is an object detection model that builds upon the foundations of previous R-CNN (Region-based Convolutional Neural Networks) architectures, offering improved speed and accuracy. The model was proposed in [30].

This model follows the next components: convolutional backbone, region proposal network, ROI pooling, object classification and bounding box regression.

The backbone of Faster R-CNN is a deep convolutional neural network (CNN) that serves as a feature extractor. Commonly used architectures include VGG-16, ResNet-50, or ResNet-101, which are pre-trained on large datasets like ImageNet. The convolutional backbone processes the input image to produce a feature map that retains spatial hierarchies and visual patterns from the image. This feature map is a compressed representation of the input image, capturing essential details necessary for object detection. Typically, the backbone consists of several convolutional layers followed by max-pooling layers. These layers progressively reduce the spatial dimensions of the image while increasing the depth of the feature maps, which represent increasingly abstract features.

The RPN is the core innovation in Faster R-CNN, responsible for generating region proposals directly from the feature map produced by the convolutional backbone. The RPN is a fully convolutional network, meaning it operates on the entire feature map without any fully connected layers. The RPN uses a sliding window approach, where a small network (e.g., a 3x3 convolutional layer) slides over the feature map. For each location in the feature map, the RPN generates multiple anchors. Anchors are predefined bounding boxes with different aspect ratios and scales that act as reference points for generating region proposals. Typically, three aspect ratios (e.g., 1:1, 1:2, 2:1) and three scales (e.g., 128x128, 256x256, 512x512) are used, resulting in nine anchors per location on the feature map. These anchors help cover a wide

range of object shapes and sizes. For each anchor, the RPN predicts two features:

1. Objectness Score: The likelihood that the anchor contains an object versus background.

2. Bounding Box Regression: Adjustments (offsets) to the anchor's coordinates to better fit the actual object.

The RPN outputs a set of proposals (bounding boxes) with their associated objectness scores. To reduce redundancy, non-maximum suppression is applied to filter out overlapping proposals, retaining only those with the highest objectness scores.

Once the RPN generates region proposals, these proposals are mapped onto the feature map to extract fixed-size feature maps for each proposal. This process is known as RoI Pooling. Since the proposals can have varying sizes, RoI Pooling ensures that each one results in a fixed-size output (e.g., 7x7) by dividing it into equal-sized regions and applying max pooling within each region. RoI Pooling allows the network to handle proposals of different sizes and shapes uniformly, making it possible to use a fully connected layer for classification.

The fixed-size feature maps produced by RoI Pooling are fed into a series of fully connected layers for object classification and bounding box regression. A softmax layer is used to classify the content of each region into one of the pre-defined object classes (e.g., person, car, dog) or background. In addition to classifying the objects, the network also predicts offsets to refine the bounding box coordinates further, making the predictions more precise.

3.2.2 Single Shot Detection

The SSD algorithm is composed of a single deep neural network that performs both object localization and classification simultaneously. The network is typically based on a backbone convolutional neural network (CNN) for feature extraction, followed by several convolutional layers that predict object classes and bounding box offsets.

SSD uses a pre-trained CNN like VGG16, MobileNet, or ResNet as the backbone for feature extraction. The earlier layers of the backbone network extract low-level features such as edges, textures, and patterns, while the deeper layers capture higher-level semantic information.

Unlike traditional detectors, SSD detects objects at multiple scales by applying small convolutional filters to multiple feature maps of different sizes. These feature maps are produced by the backbone network and additional convolutional layers that progressively decrease in size.

SSD uses a concept called "default boxes"(also known as anchor boxes) at each location in the feature maps. Default boxes are pre-defined bounding boxes of various aspect ratios and scales, tiled across the image. Each default box is associated with specific feature map cells and is responsible for detecting objects with similar shapes and sizes.

At each location on the feature maps, SSD predicts the confidence scores for each class (including a background class) for every default box. This prediction is made by applying a small convolutional filter (usually 3x3) that outputs the class scores.

Simultaneously, SSD predicts the offsets required to adjust the default boxes to better fit the actual objects. This involves predicting four values: center-x, center-y, width, and height offsets relative to the default box where x and y are the offsets calculated.

3.2.3 YOLOv8

YOLOv8 builds upon the advancements made in previous YOLO models, particularly YOLOv5, while introducing several improvements in terms of architecture, efficiency, and performance. Like its predecessors, YOLOv8 is designed for real-time object detection, making it suitable for applications where quick decision-making is critical, such as autonomous vehicles, surveillance, and robotics.

The backbone network in YOLOv8 is responsible for extracting features from the input image. YOLOv8 typically uses a deep CNN as its backbone, which has been optimized for better feature extraction. YOLOv8 utilizes CSPDarknet53, a modified version of the Darknet architecture. This modification incorporates Cross Stage Partial networks, enhancing the learning capacity and efficiency.

The neck network consists of layers that further process the features extracted by the backbone.YOLOv8 introduces PANet (Path Aggregation Network), a feature pyramid network that facilitates information flow across different scales. PANet enhances the model's ability to handle objects with diverse scales in a more effective manner.

3.3 License Plate Detection

The second stage of the system focuses on detecting the license plates within the cropped vehicle images generated from the first stage. This step is crucial as it isolates the region of interest for further processing in the character recognition stage. For this task, two approaches were implemented, both using the YOLOv8 model: object detection and segmentation.

3.3.1 YOLOv8 Object Detection

For detecting the license plates, a pretrained **YOLOv8** model to detect license plates. YOLOv8 was chosen for its speed, accuracy, and ability to detect objects at different scales, making it particularly suited to handle license plates on vehicles of varying sizes, orientations, and lighting conditions.

The YOLOv8 model was initially trained on the COCO dataset, but was retrained to detect license plates.

The detection process begins by passing the cropped vehicle images from the first stage to the YOLOv8 model, which predicts bounding boxes around the detected license plates. These bounding boxes are then used to crop the license plate region from the vehicle image.

3.3.2 YOLOv8 Segmentation

In addition to object detection, a segmentation approach using **YOLOv8 Segmentation** was implemented to achieve more precise localization of the license plates compared to the detection model. This segmentation model, unlike the object detection variant, is capable of generating a pixel-wise mask of the license plate, capturing its exact boundaries rather than just a bounding box. This approach is particularly useful in cases where the license plate is not rectangular or is partially obscured by surrounding objects or lighting conditions.

Segmentation also provides an added benefit for the subsequent character recognition stage. Since the pixel-wise mask captures the exact boundaries of the license plate, it allows for the correction of any skew or perspective distortion in the license plate region. This preprocessing step ensures that the plate is properly aligned before character recognition, thereby improving the accuracy of optical character recognition (OCR) in detecting and reading the alphanumeric content.

The YOLOv8 segmentation model was trained on the dataset described in Chapter 3.5.2, where each image was annotated with segmentation masks, marking the exact pixels of the license plates.

For the model configuration, a pretrained YOLOv8 model was used. The model, initially trained on the COCO dataset. The architecture was modified to focus solely on detecting license plates as a single class. During training, the AdamW optimizer was used with a learning rate of 0.002 and momentum of 0.9, and the model was trained for 20 epochs.

3.4 Character Recognition

The third and final stage of the system is responsible for recognizing and extracting the alphanumeric characters from the detected license plate regions. This task is crucial because the goal of the system is to convert the visual information from the license plate into machine-readable text, which can be used for various applications such as vehicle identification, access control, or law enforcement. For the character recognition task, the open-source **Tesseract OCR** engine was utilized.

3.4.1 Tesseract OCR

Tesseract [31] is an open-source OCR engine developed by Hewlett-Packard in the 1980s and later improved by Google. OCR technology enables machines to recognize and extract text from images, and Tesseract is one of the most widely used OCR tools due to its versatility and strong performance across various languages and character sets.

Tesseract operates by taking an image, analyzing the shapes and patterns it detects, and attempting to convert them into machine-readable text.

The process Tesseract follows for OCR can be summarized as:

1. **Preprocessing:** Before Tesseract performs character recognition, the input image undergoes a series of preprocessing steps, such as binarization (converting the image to black-and-white), noise reduction, and sometimes skew correction (which is handled by the segmentation model discussed earlier). These steps are crucial for improving Tesseract's performance, as it relies on clean, high-contrast input to accurately identify text.

2. **Character Segmentation:** Once the image is preprocessed, Tesseract performs character segmentation, where it divides the image into individual characters. It does this by analyzing the spaces between characters and identifying the boundaries of each one.

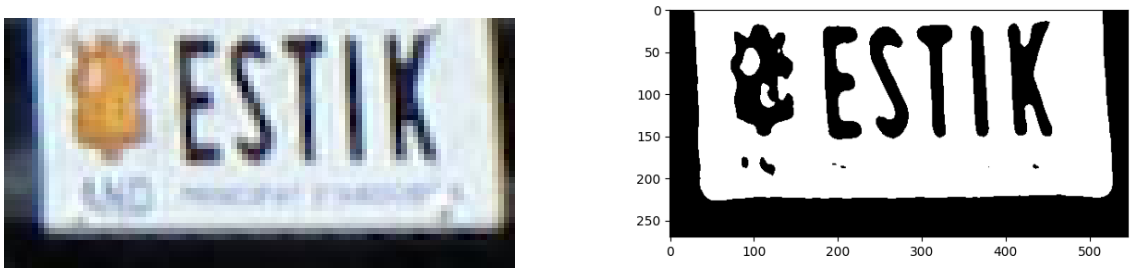
3. **Feature Extraction and Recognition:** After segmenting the characters, Tesseract extracts features from each character, such as its shape, structure, and relative position. It then matches these features to a predefined database of character templates. Using this comparison, Tesseract identifies the best match and assigns it to the corresponding character in the output string.

3.4.2 Preprocessing for Improved OCR Accuracy

In this system, several preprocessing techniques were applied to the license plate image before passing it to Tesseract to maximize its recognition accuracy. These

include:

- **Grayscale Conversion:** The input image is converted to grayscale because usually license plates are black and white so there is no need for color information.
- **Binarization:** The grayscale image is thresholded to produce a binary image (black and white), which allows Tesseract to better differentiate between the characters (foreground) and the background.
- **Noise Removal:** Morphological operations and filters are applied to remove unwanted noise, such as small dots or specks, that might interfere with character recognition.
- **Skew Correction:** The segmentation model discussed in the previous stage is particularly useful here, as it ensures that the license plate region is aligned properly, with minimal skew. This step is critical for improving the accuracy of character recognition since tilted or misaligned text can lead to incorrect OCR results.



(a) *Before image processing* (b) *After image processing*

Figure 3.2: *Result of image processing in incense plate.*

3.5 Datasets

The success of any deep learning-based object detection and recognition system largely depends on the quality and diversity of the datasets used for training and evaluation. This section will detail the datasets employed for training, evaluating, and testing the models in this project.

3.5.1 COCO Dataset

For this project, pretrained models (Faster R-CNN, SSD, and YOLOv8) were used, which were initially trained on the COCO dataset [28]. The COCO dataset includes classes like car, bus, truck, and motorcycle, which are particularly relevant to the task of vehicle detection and classification in this license plate recognition system.

Although the dataset doesn't specifically focus on license plates, its vast diversity and comprehensive object annotations make it a valuable resource for fine-tuning models to handle complex scenarios such as object occlusion and varying object sizes in vehicle detection.

3.5.2 OpenImages Dataset

To evaluate the vehicle detection and classification models, as well as license plate detection, the OpenImages [29] dataset was used. OpenImages is a large-scale visual dataset containing around 9 million images annotated with image-level labels, object bounding boxes, and visual relationships. It is an excellent resource for tasks involving object detection, as it provides annotations for thousands of object classes in real-world environments.

For this project, a test subset of 2,158 images was downloaded, where each image includes at least one annotation of a vehicle such as car, bus, truck, and motorcycle, and at least one annotation of a visible license plate. The diversity of vehicle types and environmental settings in the OpenImages dataset ensures robust testing of the vehicle detection models, helping assess their performance in real-world scenarios where license plates may be partially occluded or in various orientations.

This subset was created to ensure that all images used for evaluation contain vehicles with license plates, allowing for the simultaneous assessment of vehicle detection, classification, and license plate detection.

It was also created a train subset of 2,905 images to train the YOLOv8 model to segment license plates.

Examples of class Car.

Examples of class Motorcycle.

Examples of class Truck.

Examples of class Bus.

Figure 3.3: *Examples of images of each class for the OpenImages Dataset.*

3.5.3 Platesmania Dataset

In addition to the OpenImages dataset, a custom dataset from Platesmania was created, a website that contains a vast collection of images of vehicles with visible license plates, primarily from across Europe. This dataset includes not only the images but also the associated metadata, such as the alphanumeric string representing the license plate, the model, and the brand of the vehicle.

The Platesmania dataset was crucial for the character recognition stage of the project. Unlike other datasets, it provides ground-truth labels for the text on the license

plates, which can be used for training and evaluating character recognition models. The diversity in license plate formats, fonts, and lighting conditions across various European countries also ensures that the model can generalize well to different styles of plates. Furthermore, the inclusion of metadata such as vehicle make and model provides opportunities for deeper analysis or potential model enhancements in future work.

By combining publicly available datasets like COCO and OpenImages with a custom-curated dataset from Platesmania, this project leverages both general-purpose and domain-specific data to achieve comprehensive evaluation and testing of the license plate detection and recognition system.

3.6 System Evaluation Metrics

To evaluate the performance of the License Plate Detection and Recognition System, several metrics were employed. These metrics provide a comprehensive view of the model's accuracy, precision, recall, and overall effectiveness.

3.6.1 Intersection over Union

The IoU metric is fundamental in object detection tasks, as it measures how well the predicted bounding boxes overlap with the ground truth bounding boxes. IoU provides a clear measure of how close the predicted object localization is to the actual object, making it crucial in evaluating object detection models. It is calculated as:

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{3.1}$$

Area of Overlap: The region where the predicted and ground truth bounding boxes overlap.
Area of Union: The total area covered by both the predicted and ground truth bounding boxes.

An IoU of 1.0 indicates perfect overlap, meaning the predicted bounding box exactly matches the ground truth, while an IoU of 0 indicates no overlap at all.

High IoU Threshold:

- **Definition:** A high IoU threshold (e.g., 0.75 or 0.9) requires a greater degree of overlap between the predicted and ground truth bounding boxes for a detection to be considered a true positive.

- **Implications:**

  - *Precision:* A higher threshold typically results in higher precision because the model must make very accurate predictions to achieve a high IoU, reducing the number of false positives.
  - *Stringency:* The high threshold sets a stricter criterion for what constitutes a correct detection. This can be beneficial for tasks requiring very accurate localization but might lead to fewer detections being classified as true positives.

Low IoU Threshold:

- **Definition:** A low IoU threshold (e.g., 0.5) requires less overlap between the predicted and ground truth bounding boxes for a detection to be considered a true positive.
- **Implications:**
  - *Recall:* A lower threshold generally results in higher recall because more detections are classified as true positives due to the reduced overlap requirement, capturing more instances of the object.
  - *Flexibility:* The low threshold is less stringent, making it easier for predictions to qualify as correct.

In this project, an IoU threshold of 0.5 and 0.75 were used to balance between precision and recall, ensuring that the model detects objects effectively while maintaining reasonable localization accuracy.

3.6.2 Precision and Recall Curves

Precision-Recall Curves provide a graphical representation of the trade-off between precision and recall for each class, such as cars, buses, trucks, motorcycles, and license plates. These curves give insights into how the model performs at different confidence thresholds.

Figure 3.4: *Visual representation of IoU.*

Precision: The ratio of true positive detections to the total number of detections (true positives + false positives).

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

Recall: The ratio of true positive detections to the total number of ground truth objects.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

These curves help evaluate the model's ability to balance precision and recall for each class, allowing for targeted improvements in detecting certain types of objects.

3.6.3 Mean Average Precision (mAP)

The mean Average Precision (mAP) is a standard metric used for evaluating object detection systems across multiple classes. mAP aggregates precision values at different IoU thresholds and provides a single value that summarizes the model's performance. Specifically, mAP measures the ability of the system to detect objects accurately while minimizing false positives.

The formula for mAP is:

$$mAP = \frac{1}{n} \sum_{i=1}^{n} AP(i) \tag{3.4}$$

where $AP(i)$ represents the Average Precision for each class i, and n is the total number of classes (e.g., car, bus, truck, motorcycle, license plate).
In this project, AP is calculated for each vehicle type and license plate at varying IoU thresholds (e.g., IoU = 0.5, 0.75, 0.9).

3.6.4 Mean Average Recall (mAR)

While mAP focuses on precision, Mean Average Recall (mAR) measures how effectively the model retrieves relevant objects. Recall is defined as the ratio of true positives (correctly detected objects) to the total number of ground truth objects. mAR measures the ability of the model to detect all objects, regardless of how precise the bounding boxes are.

The formula for mAR is:

$$mAR = \frac{1}{n} \sum_{i=1}^{n} AR(i) \tag{3.5}$$

where $AR(i)$ is the Average Recall for class i, and n is the number of object classes. A high mAR indicates that the model is good at detecting most objects (high recall), but it may also mean the model is less precise, resulting in more false positives.

3.6.5 Character Error Rate (CER)

In the context of Optical Character Recognition (OCR), the Character Error Rate (CER) measures the accuracy of the predicted text from license plates. CER is a standard metric in text recognition tasks and is calculated as:

$$CER = \frac{\text{Substitutions} + \text{Insertions} + \text{Deletions}}{\text{Total Characters in Ground Truth}} \tag{3.6}$$

Substitutions: Incorrect characters in place of the correct one.
Insertions: Extra characters not present in the ground truth.
Deletions: Characters missed by the OCR system.

CER was used as the primary metric for evaluating the OCR's performance in this project, with efforts made to minimize it through data quality improvements and model fine-tuning

4 Results

This chapter presents the evaluation of the proposed license plate detection and recognition system based on the results obtained through various experiments. The system was tested across three main stages. For each stage, the performance is systematically evaluated using key metrics, such as mean mAP and CER to measure the accuracy and reliability of the models used.

4.1 Stage1: Vehicle Detection and Classification

The vehicle detection and classification stage is crucial for isolating the vehicles within an input image, classifying them into predefined categories—car, bus, truck, or motorcycle—and passing them forward for license plate detection. In this section, the performance of the three object detection models used: Faster R-CNN, SSD, and YOLOv8 are compared. The models are evaluated using key metrics such as precision-recall curves, AP per class, mAP, AR per class and mAR to better understand their strengths and weaknesses.

All models were tested on the test subset of OpenImagesDataset, mentioned in 3.5.2 containing 2,158 images. Two thresholds of IoU were tested 0.50 and 0.75.

4.1.1 Faster R-CNN

The performance of the Faster R-CNN model for vehicle detection and classification in Stage 1 was evaluated using PR curves and AP and AR metrics. The results for the four vehicle classes (Cars, Motorcycles, Buses, and Trucks) are presented in Figures 4.1 and 4.2 (PR curves for IoU at 0.50 and 0.75 respectively) and Table 4.1 (AP and AR).

With IoU at 0.5, Figure 4.1, the model demonstrates the strongest performance in detecting buses, with consistently high precision and recall across most thresholds.

For cars and motorcycles, the model shows moderate precision-recall performance.

Both curves reveal a gradual decline in precision as recall increases. This indicates that, although the model can recall a significant portion of these vehicles, there is a slight decrease in precision that means, more false positives are introduced as recall improves.

In contrast, truck detection presents the greatest challenge, as shown by the steep decline in precision in the red curve. As recall increases, precision drops sharply, especially at higher recall values, meaning the model generates a larger number of false positives when identifying trucks.

With IoU at 0.75, Figure 4.2, the model loses in precision, especially in the class cars, that drops significantly at lower recall values. For the other classes, although the curves decline sooner than with IoU=0.5, they still maintain good performance this indicates that the model is accurate in terms of the location of the objects.

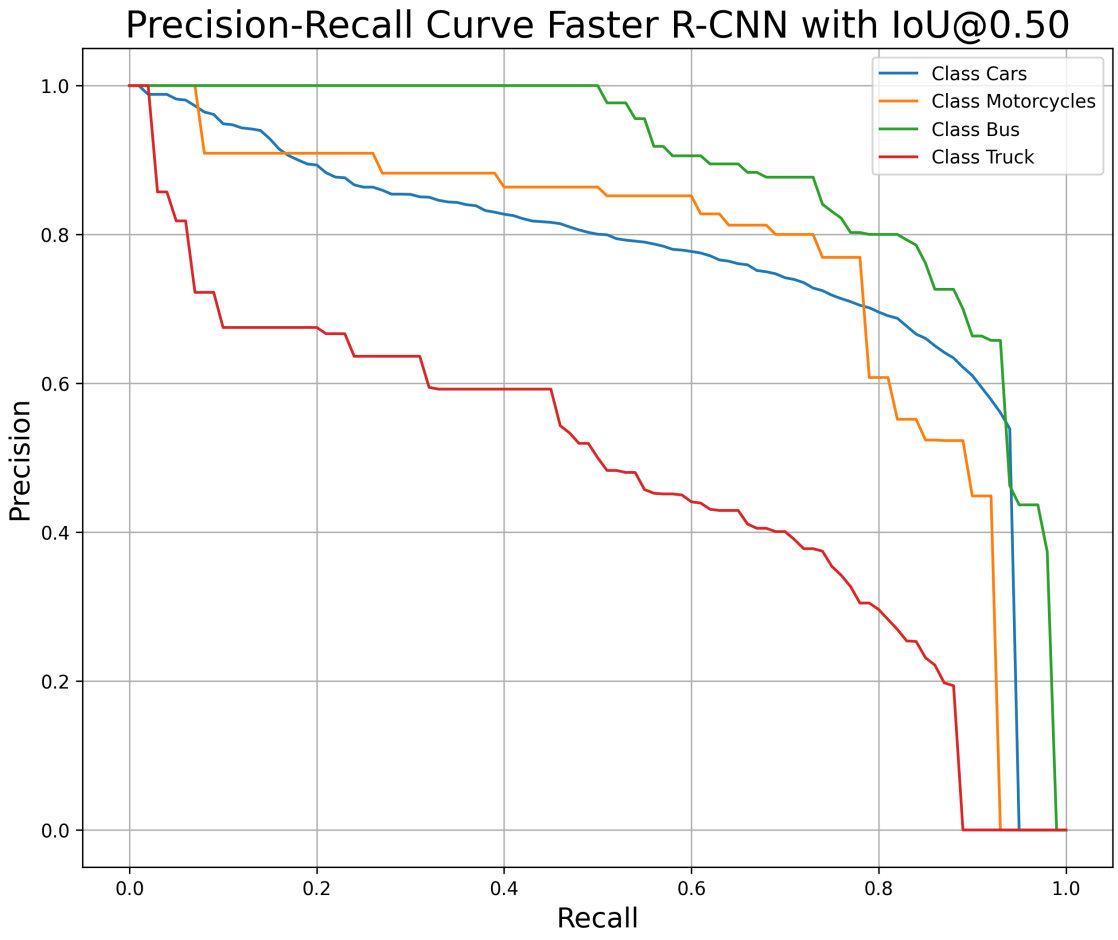


Figure 4.1: *Faster R-CNN precision-recall curves of each class with IoU=0.50.*

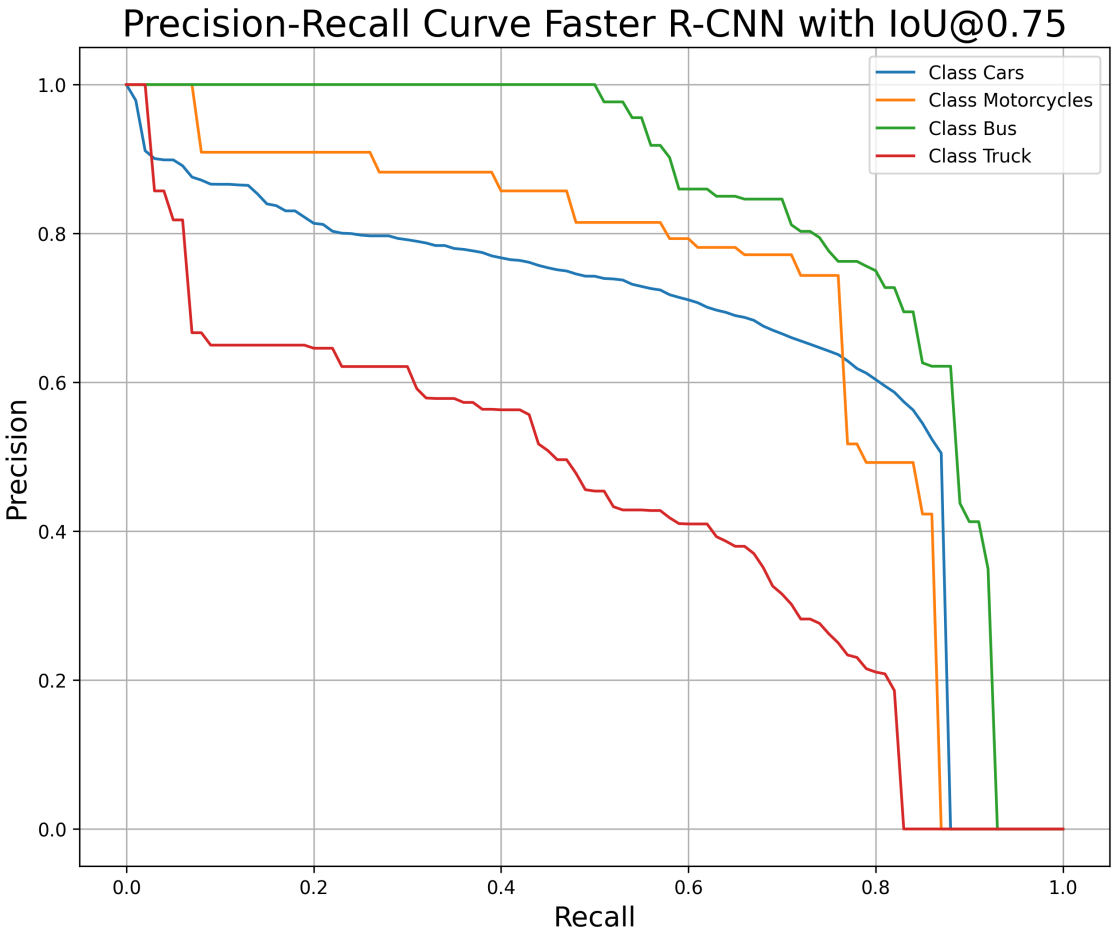


Figure 4.2: *Faster R-CNN precision-recall curves of each class with IoU=0.75.*

The AP and AR values for each class, shown in Table 4.1, further highlight the model's performance across the four vehicle classes. The bus class achieves the highest AP (0.89 and .83) and AR (0.99 and 0.93) at both thresholds of IoU, confirming its superior detection performance. The car class is the only one that drops its performance significantly when the IoU is more strict, as it was said above, but still maintains an higher AR (0.88) than the class motorcycle (0.87) and truck (0.83). Apart for the class car all others do not drop in performance as the IoU gets stricter.

	IoU@0.5		**IoU@0.75**	
Class	**AP**	**AR**	**AP**	**AR**
Car	0.76	0.95	0.66	0.88
Motorcycle	0.76	0.92	0.71	0.87
Bus	0.89	0.99	0.83	0.93
Truck	0.48	0.88	0.43	0.83
	0.72	0.93	0.66	0.88

Table 4.1: AP and Average Recall AR for Faster R-CNN at IoU thresholds of 0.5 and 0.75

4.1.2 YOLOv8

The performance of YOLOv8 model for vehicle detection and classification in Stage 1 was evaluated using PR curves and AP and AR metrics. The results for the four vehicle classes are presented in Figures 4.3 and 4.4 (PR curves for IoU at 0.50 and 0.75 respectively) and Table 4.2 (AP and AR).

Similar to the Faster R-CNN model, the YOLOv8 model demonstrates the strongest performance in detecting buses, with consistently high precision and recall at both IoU thresholds.

For cars the model loses performance as IoU gets stricter, like in the Faster R-CNN. As for the other classes, the curve is similar in both IoU thresholds.

Truck detection remains the most challenging task, as shown by the steep decline in precision in the red curve. As recall increases, precision drops sharply, especially at higher recall values, meaning the model generates a larger number of false positives when identifying trucks.

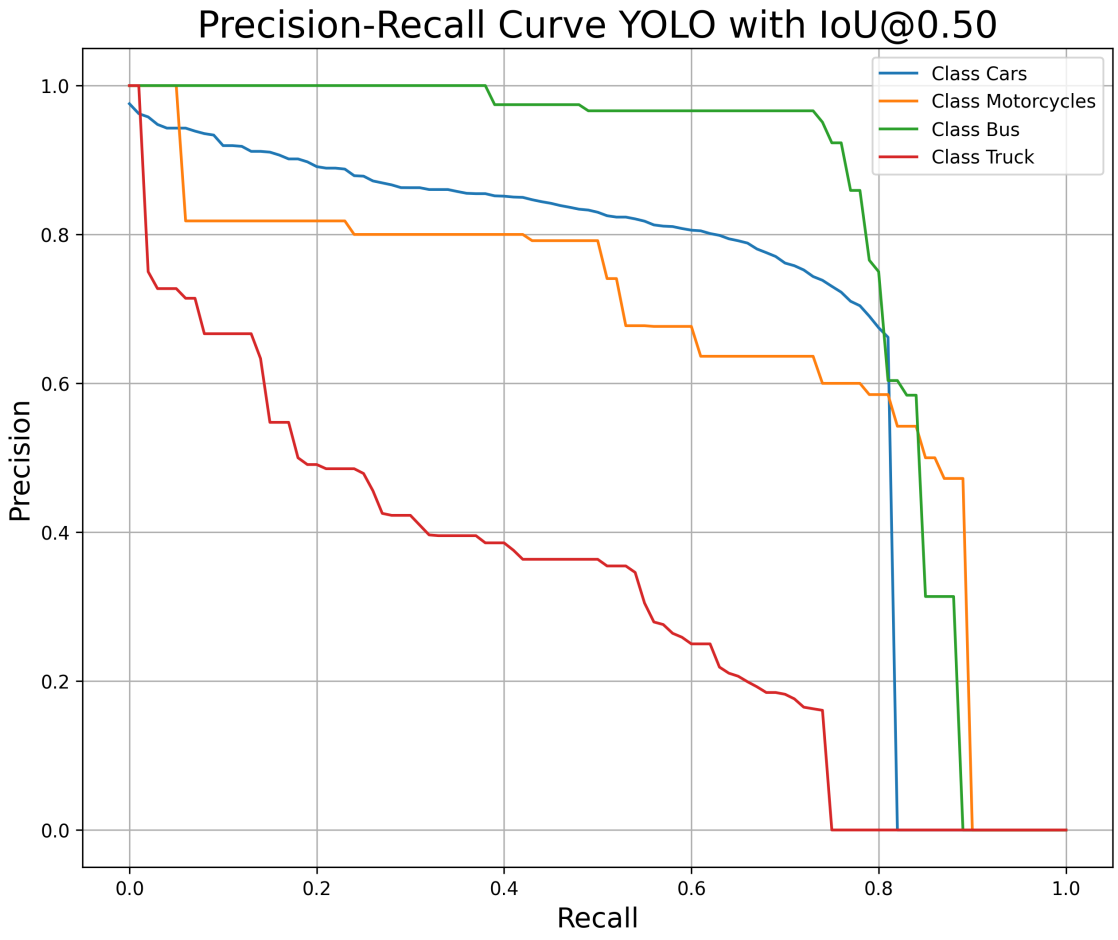


Figure 4.3: *YOLOv8 precision-recall curves of each class with IoU=0.50.*

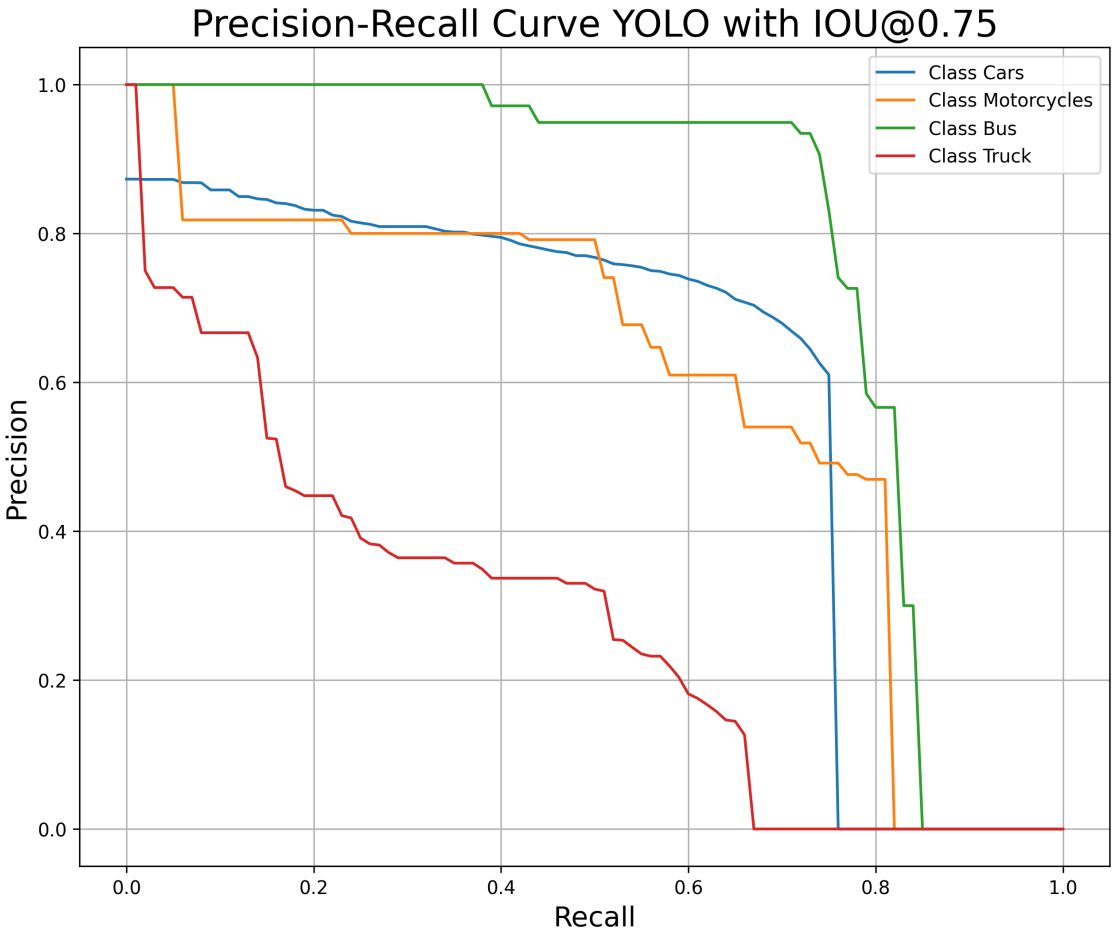


Figure 4.4: *YOLOv8 precision-recall curves of each class with IoU=0.75.*

The AP and AR values for each class, are shown in Table 4.2. The bus class achieves the highest AP (0.89 and .83) and AR (0.79 and 0.85) at both thresholds of IoU. The classes car and motorcycle drop its performance significantly when the IoU is more strict. The mAP of the model drops from 0.62 to 0.56 and the mAR from 0.93 to 0.77.

	IoU@0.5		**IoU@0.75**	
Class	**AP**	**AR**	**AP**	**AR**
Car	0.68	0.82	0.59	0.76
Motorcycle	0.68	0.89	0.60	0.82
Bus	0.82	0.88	0.79	0.85
Truck	0.32	0.75	0.28	0.67
	0.62	0.93	0.56	0.77

Table 4.2: AP and AR for YOLOv8 at IoU thresholds of 0.5 and 0.75

4.1.3 SSD

The performance of SSD for vehicle detection and classification in Stage 1 was evaluated using PR curves and AP and AR metrics. The results for the four vehicle classes are presented in Figures 4.5 and 4.6 (PR curves for IoU at 0.50 and 0.75 respectively) and Table 4.3 (AP and AR).

At both IoU thresholds of 0.5 and 0.75, the performance of the bus and truck classes remains relatively stable. Among these, the bus class exhibits the highest performance, while the truck class shows the lowest. Notably, at an IoU threshold of 0.75, the precision for car and motorcycle classes declines significantly at lower recall values.

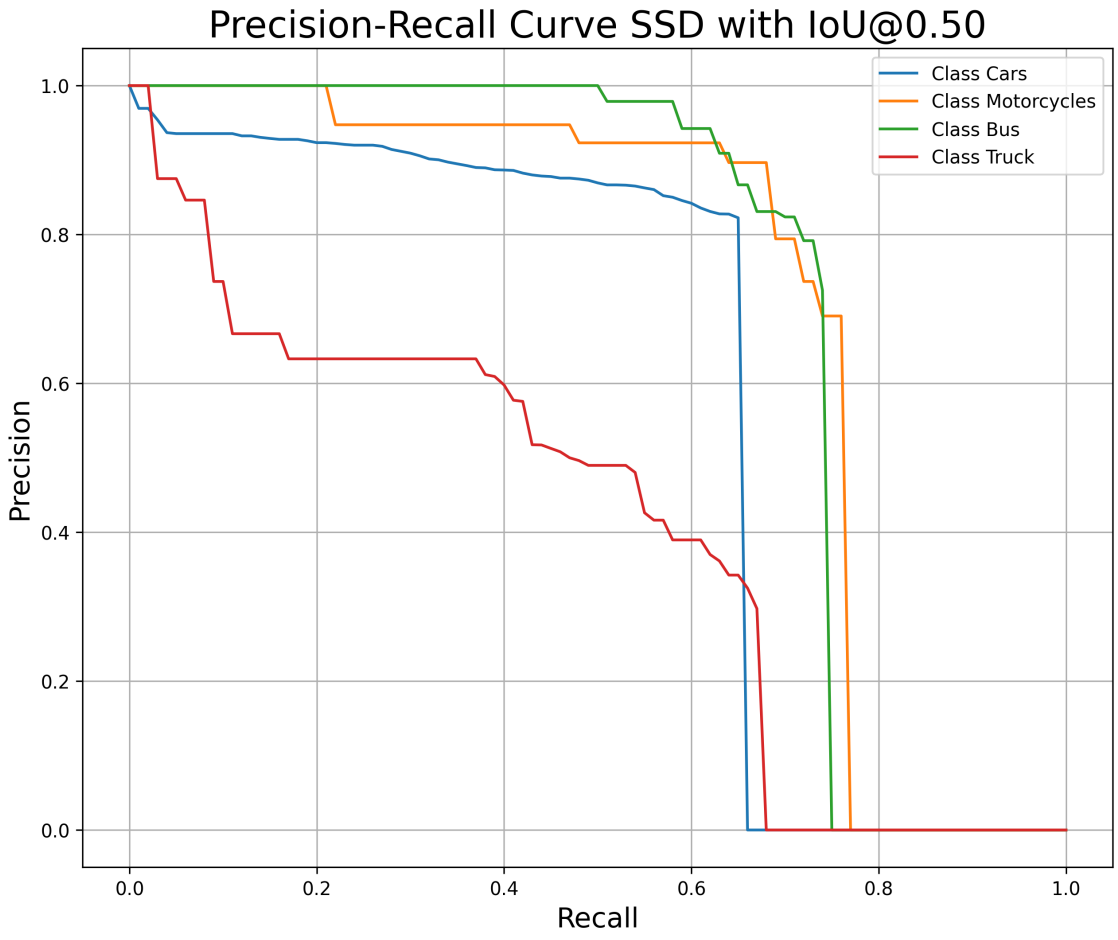


Figure 4.5: *SSD precision-recall curves of each class with IoU=0.50.*

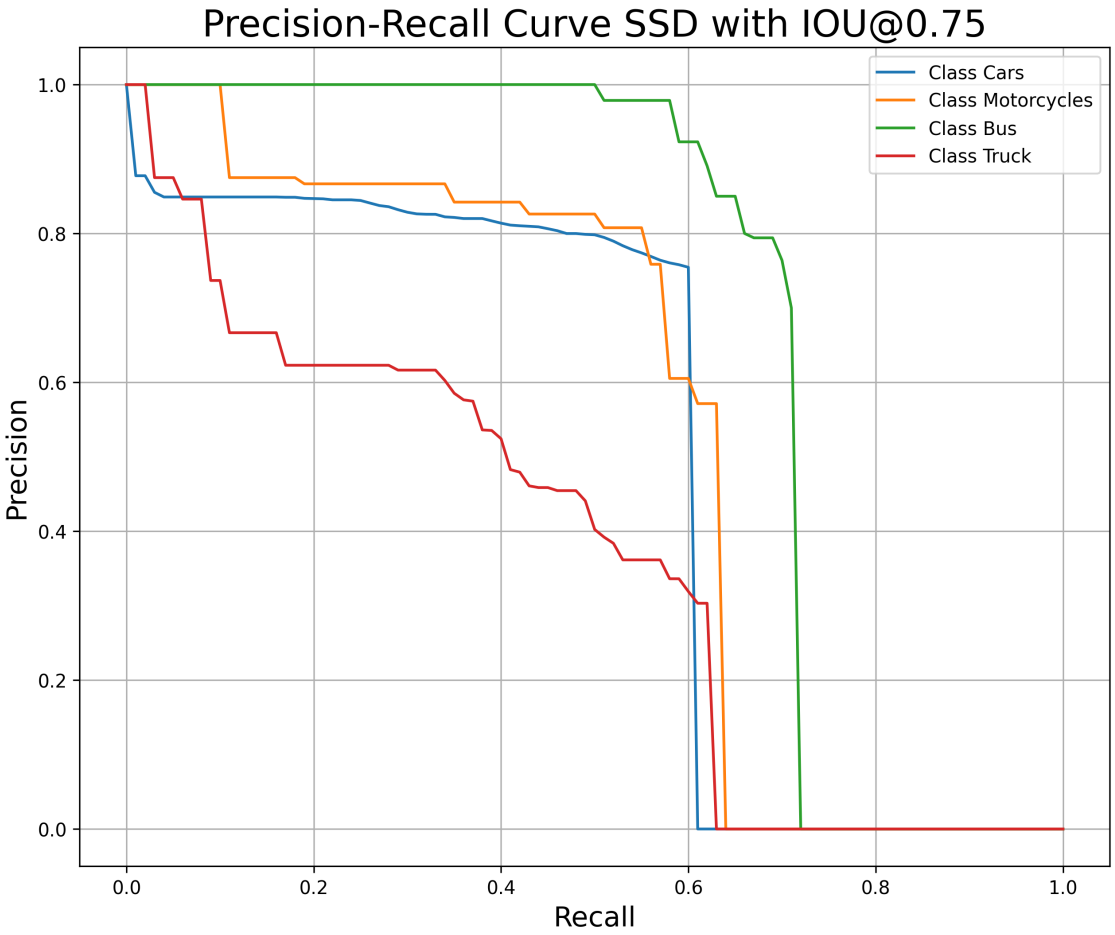


Figure 4.6: *SSD precision-recall curves of each class with IoU=0.75.*

The AP and AR values for each class, shown in Table 4.3. The bus class demonstrates remarkable stability, with AP values of 0.72 and 0.70, and AR values of 0.74 and 0.72 at IoU thresholds of 0.5 and 0.75, respectively. This indicates that the model consistently detects buses with high precision and recall, regardless of the IoU threshold.

In contrast, the car and motorcycle classes exhibit significant drops in both AP and AR values when the IoU threshold increases from 0.5 to 0.75. For cars, the AP decreases from 0.58 to 0.50, and the AR from 0.65 to 0.60. Similarly, for motorcycles, the AP drops from 0.71 to 0.54, and the AR from 0.76 to 0.63. These declines suggest that the model's ability to accurately detect and classify cars and motorcycles diminishes as the overlap requirement becomes more stringent.

The truck class consistently shows the lowest performance among the four classes, with AP values of 0.40 at IoU 0.5 and 0.36 at IoU 0.75, and AR values of 0.67 and 0.63, respectively. This indicates that the model struggles the most with detecting and classifying trucks, which could be due to the greater variability in truck appearances or the complexity of their shapes.

	IoU@0.5		IoU@0.75	
Class	**AP**	**AR**	**AP**	**AR**
Car	0.58	0.65	0.50	0.60
Motorcycle	0.71	0.76	0.54	0.63
Bus	0.72	0.74	0.70	0.72
Truck	0.40	0.67	0.36	0.63
	0.60	0.71	0.52	0.64

Table 4.3: AP and Average Recall AR for SSD at IoU thresholds of 0.5 and 0.75

4.1.4 Ensemble Method

Ensemble learning methods aim to improve predictive performance by combining the outputs of multiple models. The intuition behind this approach is that different models may be better in different scenarios, thus providing complementary information when making predictions. By aggregating the predictions from Faster R-CNN, YOLOv8, and SSD, the strengths of each model can be leveraged to obtain a more robust final output.

In this experiment, Weighted Box Fusion (WBF) was applied to combine predictions from three object detection models: Faster R-CNN, YOLOv8, and SSD. WBF is a technique that fuses overlapping bounding boxes predicted by different models, leveraging the confidence scores to produce a single, refined bounding box for each detected object. The primary goal of using WBF was to enhance the overall detection performance by incorporating the strengths of each model.

1. **Aggregating Predictions**: Each model (Faster R-CNN, YOLOv8, SSD) detects objects independently and outputs a set of bounding boxes, class labels, and confidence scores. The bounding boxes are in the format [x1, y1, x2, y2], where $(x1, y1)$ is the top-left corner and $(x2, y2)$ is the bottom-right corner.

2. **Grouping Overlapping Boxes**: Bounding boxes that overlap significantly (based on an IoU threshold, typically 0.5) are grouped together. These overlapping boxes are considered to represent the same object.

3. **Fusing Boxes**: For each group of overlapping boxes, the final bounding box is computed as a weighted average of the coordinates of the individual boxes, where the weight is determined by the confidence scores. Boxes predicted with higher confidence contribute more to the final position, resulting in a fused bounding box that reflects the combined information from multiple models.

The performance of Ensemble method for vehicle detection and classification in Stage 1 was evaluated using PR curves and AP and AR metrics. The results for the four

vehicle classes (Cars, Motorcycles, Buses, and Trucks) are presented in Figures 4.7 and 4.8 (Precision-Recall Curves) and Table 4.4 (AP and AR).

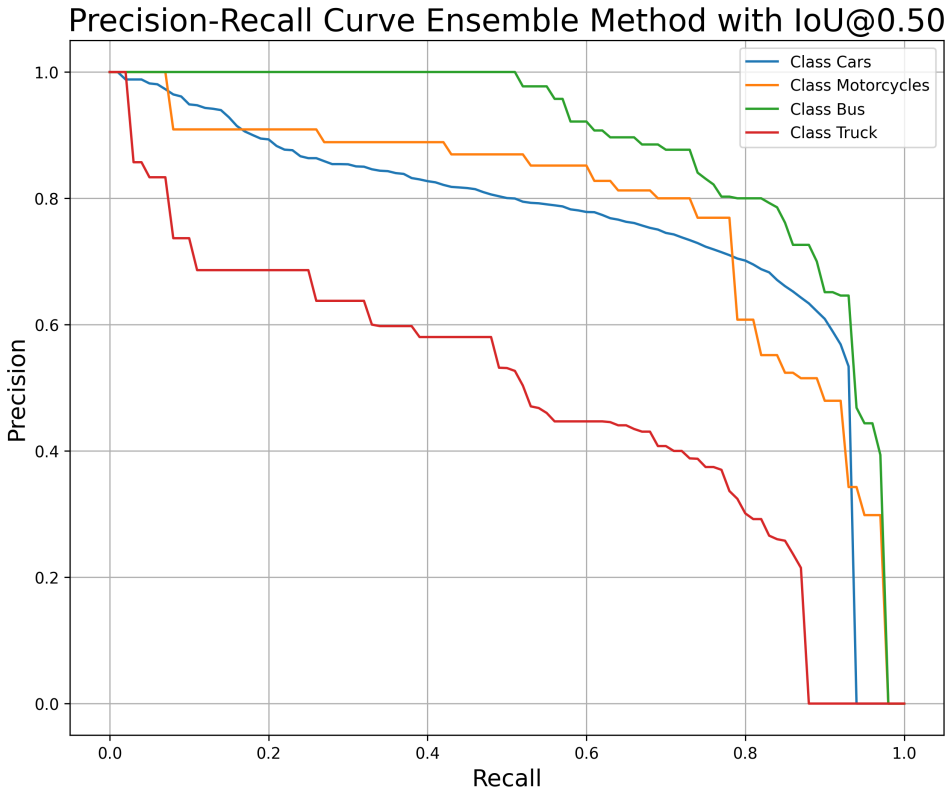


Figure 4.7: *Ensemble method precision-recall curves of each class with IoU=0.50.*

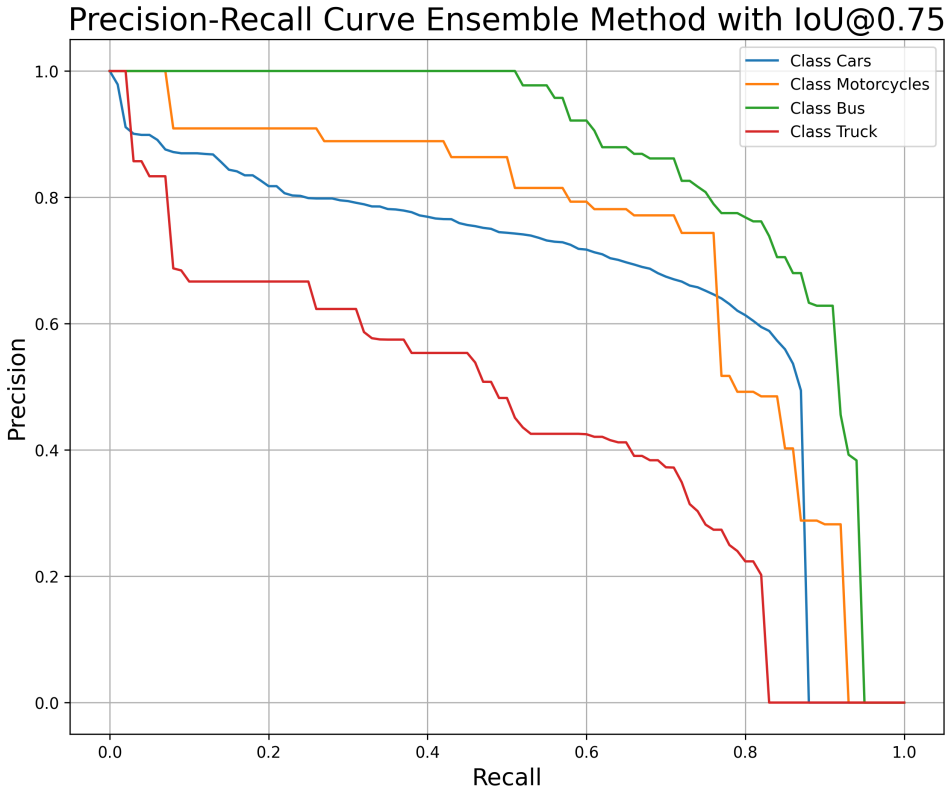


Figure 4.8: *Ensemble method precision-recall curves of each class with IoU=0.75.*

The bus class demonstrates remarkable stability, with AP values of 0.89 and 0.86, and AR values of 0.97 and 0.95 at IoU thresholds of 0.5 and 0.75, respectively. This indicates that the model consistently detects buses with high precision and recall, regardless of the IoU threshold. The car class experiences a significant drop in performance compared to the other classes when the IoU threshold increases from 0.5 to 0.75. The AP for cars decreases from 0.76 to 0.66, and the AR from 0.94 to 0.88.

	IoU@0.5		**IoU@0.75**	
Class	**AP**	**AR**	**AP**	**AR**
Car	0.76	0.94	0.66	0.88
Motorcycle	0.78	0.97	0.73	0.92
Bus	0.89	0.97	0.86	0.95
Truck	0.49	0.87	0.45	0.83
	0.73	0.94	0.67	0.89

Table 4.4: AP and Average Recall AR for ensemble method at IoU thresholds of 0.5 and 0.75

4.1.5 Models Comparison

By comparing the mAP and mAR of the models (Table 4.5) and the time each model takes to process one image (4.6), it is discuss what implications each model has to an ALPR system. Faster R-CNN achieves high accuracy, with an mAP of

Table 4.5: Table with the mAP and mAR of each model at different IoU's.

	IoU@0.5		**IoU@0.75**	
Models	**mAP**	**mAR**	**mAP**	**mAR**
Faster R-CNN	0.72	0.93	0.66	0.88
Yolov8	0.62	0.93	0.56	0.77
SSD	0.60	0.71	0.52	0.64
Ensemble Method	0.73	0.94	0.67	0.89

0.72 and an mAR of 0.93 at IoU@0.5, though its mAP drops to 0.66 at IoU@0.75. However, its processing time of 22.2 seconds per image makes it unsuitable for real-time applications.

YOLOv8, the fastest model at just 0.1 seconds per image, records an mAP of 0.62 and an mAR of 0.93 at IoU@0.5 but sees a decline in precision at stricter thresholds, with an mAP of 0.56 at IoU@0.75. This model is ideal for scenarios prioritizing speed, despite lower accuracy.

SSD offers a middle ground, processing images in 1 second with an mAP of 0.60 at IoU@0.5, which decreases to 0.52 at IoU@0.75. It balances speed and performance, making it a viable choice for certain ALPR tasks.

The Ensemble Method, which combines predictions from all three models, achieves the highest accuracy, with an mAP of 0.73 at IoU@0.5. However, the increased processing time required to run all models simultaneously renders it impractical for real-time applications.

Table 4.6: Table with the time each model takes to process one image.

Models	**Time/Image(s)**
Faster R-CNN	22.2
Yolov8	0.1
SSD	1.2

The choice of model for an ALPR system largely depends on the specific application context. For real-time applications, where speed is a critical factor, the YOLO model is often preferred due to its rapid processing capabilities, despite being slightly less precise than some alternatives. In contrast, for offline applications where processing time is less of a concern, the Faster R-CNN model can be employed effectively, as it

offers higher accuracy in detection. Additionally, ensemble methods that combine multiple models may be considered to leverage the strengths of each, improving overall performance in both detection and recognition tasks.

In table 4.6 the processing times of the images for each model were obtained using a personal computer equipped with an Intel Core i5-3230M CPU and 8 GB of RAM, without utilizing a GPU for acceleration.

4.2 Stage2: License Plate Detection

To evaluate the license plate detection performance, four types of input images were employed. The first input type was raw images, which included all images without any prior vehicle detection. The subsequent input types consisted of vehicle images detected by each of the Stage 1 models: Faster R-CNN, YOLO, and SSD. This approach allowed us to comprehensively assess the license plate detection capabilities across different input scenarios and to determine whether vehicle detection significantly improved performance.

4.2.1 Yolov8 Detection

At both IoU thresholds (as shown in Figure 4.9 and Figure 4.10) , the raw demonstrates a consistently high precision across a broad range of recall values. This sustained high precision indicates that the model is highly effective in identifying license plates while keeping the false positive rate low. The curve of the raw images remains positioned at the top for a more extended period compared to the other models.

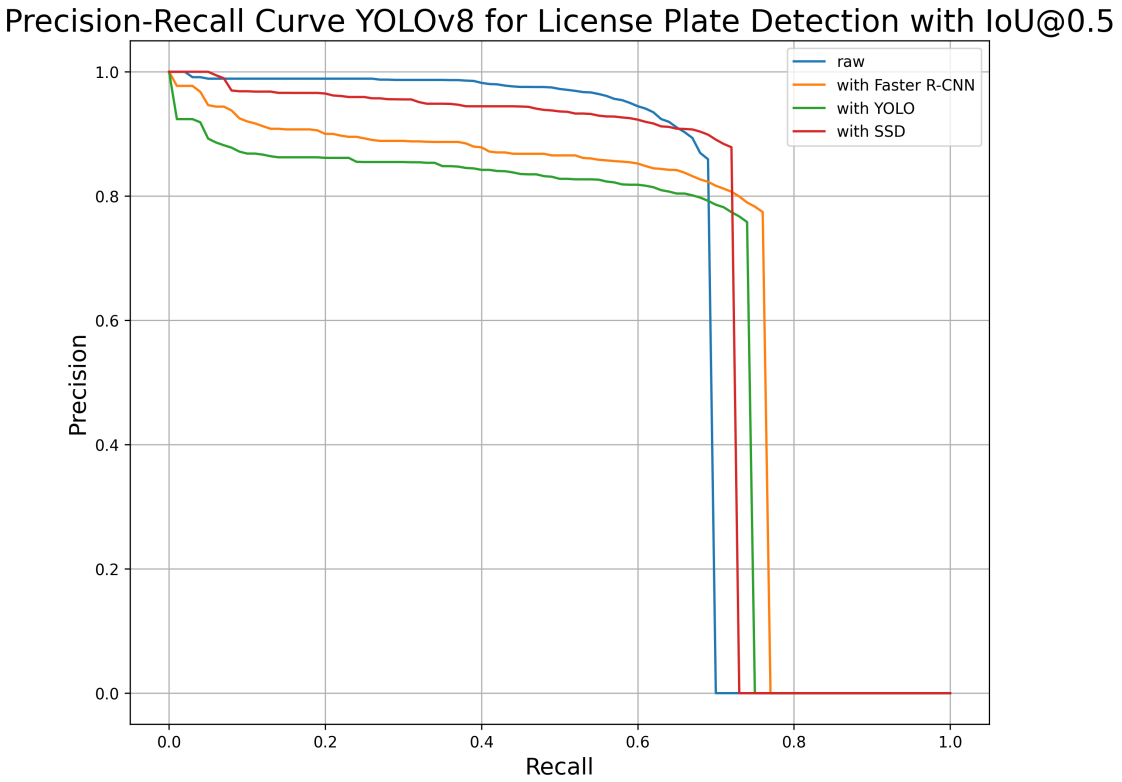


Figure 4.9: *YOLOv8 precision-recall curves with IoU=0.50 for license plate detection of each vehicle detection model.*

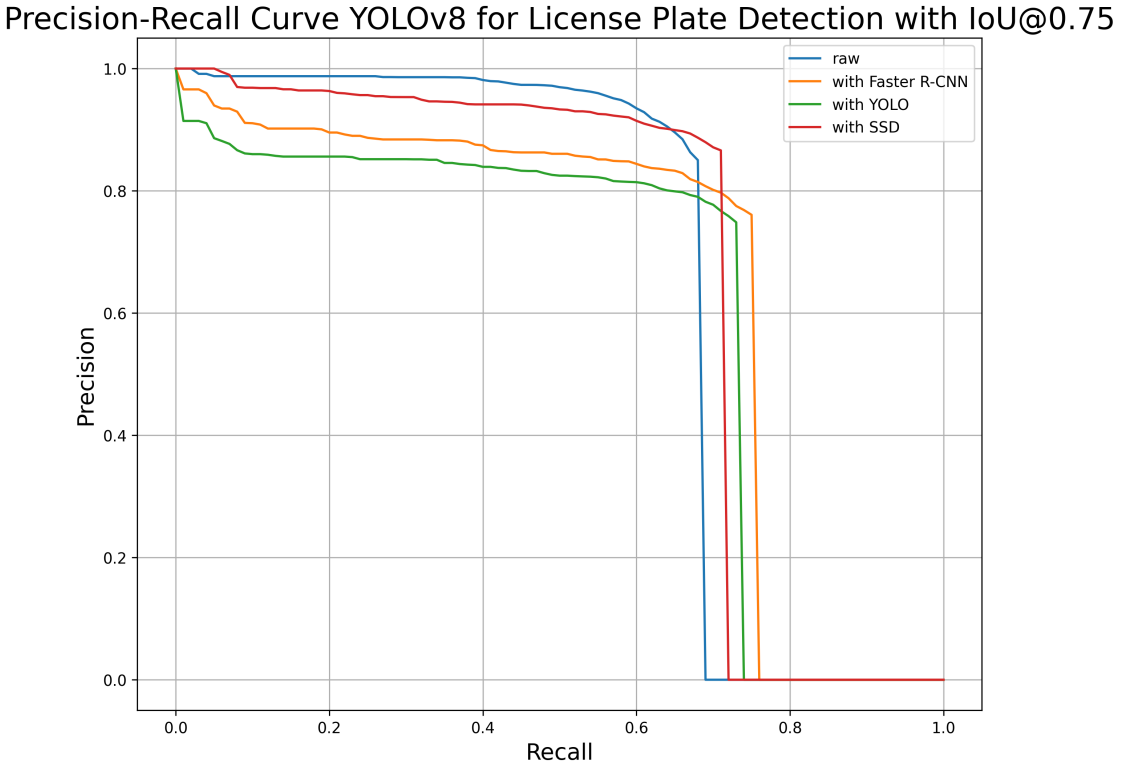


Figure 4.10: *YOLOv8 precision-recall curves with IoU=0.75 for license plate detection of each vehicle detection model.*

On the other hand Faster R-CNN as a better mAR (as shown in Table 4.7), mAR is relevant because indicates that the model is good at identifying license plates which is crucial because in an ALPR system the goal is to retrieve all license plates present in the image.

	IoU@0.5		IoU@0.75	
Class	**mAP**	**mAR**	**mAP**	**mAR**
Raw	0.67	0.69	0.66	0.68
with Faster R-CNN	0.67	0.77	0.66	0.76
with YOLOv8	0.63	0.74	0.62	0.73
with SSD	0.69	0.73	0.67	0.72

Table 4.7: AP and Average Recall AR for license plate detection of each vehicle detection model.

4.2.2 YOLOv8 Segmentation

For license plate segmentation with YOLOv8, looking at the precision-recall curves in Figure 4.11, raw images have a higher recall but it declines sooner than others curves. As for images from YOLOv8, the curve has a low precision compared to the others and also declines sooner than with images from Faster R-CNN. The curve from SSD is a good middle ground. When IoU is at 0.75 (shown in Figure 4.12) the curves are similar to the curves with IoU at .5. This means that the model provides precise boxes even as the IoU gests higher.

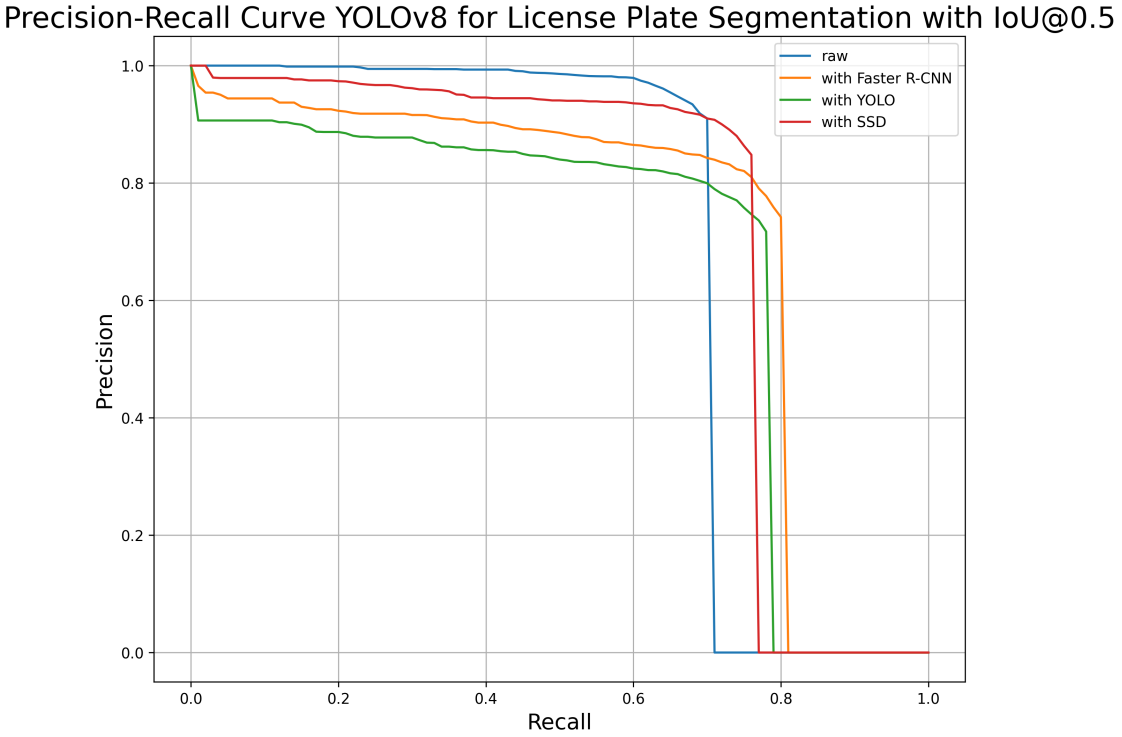


Figure 4.11: *YOLOv8 precision-recall curves with IoU=0.50 for license plate segmentation of each vehicle detection model.*

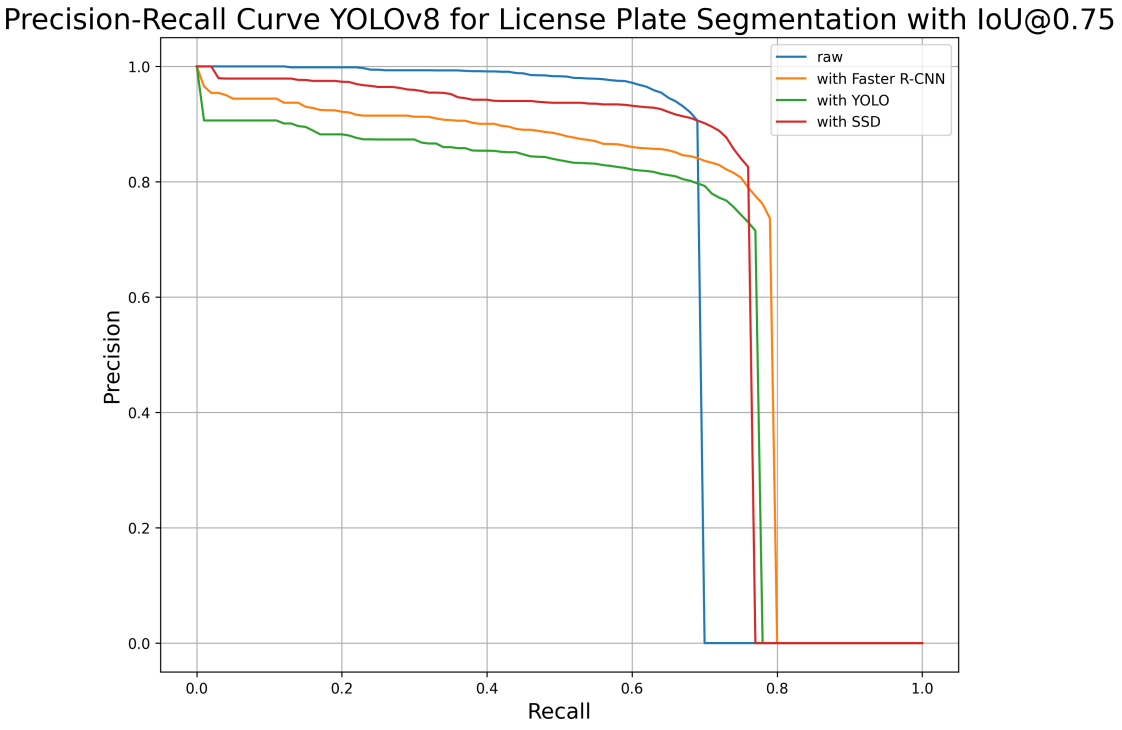


Figure 4.12: *YOLOv8 precision-recall curves with IoU=0.75 for license plate segmentation of each vehicle detection model.*

By analyzing Table 4.8 SSD is the model with best mAP at IoU=0.5 and Faster R-CNN has the best mAR. With IoU=0.75 Faster R-CNN gets a better mAP and

mAR (0.71 and 0.80 respectively).

	IoU@0.5		IoU@0.75	
Class	**mAP**	**mAR**	**mAP**	**mAR**
Raw	0.70	0.70	0.69	0.69
with Faster R-CNN	0.72	0.81	0.71	0.80
with YOLOv8	0.67	0.78	0.66	0.76
with SSD	0.73	0.77	0.71	0.76

Table 4.8: AP and Average Recall AR for license plate segmentation of each vehicle detection model.

By comparing the results presented in both tables, it is evident that the combination of Faster R-CNN with YOLO segmentation gets the best performance. This approach not only achieves one of the highest mean mAP scores but also demonstrates the best mean mAR among the evaluated models.

4.3 Stage3: Character Recognition

Character recognition was tested using two different approaches. In the first approach, the recognition was performed using the bounding box of the license plate, without any additional preprocessing. In the second approach, a correction of the orientation of the plate was applied prior to recognition. Both tests utilized a custom dataset compiled from Platesmania.

	W Orientation Correction	**W/O Orientation Correction**
CER	0.34	0.47

Table 4.9: Averge CER of Pytesseract with and without orientation correction.

The results in Table 4.9, showed that the OCR achieved a CER of 0.34 with orientation correction, compared to 0.47 without correction. The significant improvement in CER can be attributed to the alignment of characters after leveling the plates. When license plates are skewed or rotated, the OCR model has to process distorted character shapes, which increases the likelihood of errors in character identification. Orientation correction mitigates this problem by standardizing the input, ensuring that the characters are aligned horizontally, as they would typically appear in a well-captured image. This alignment reduces ambiguity and makes it easier for the OCR system to accurately interpret each character.

4.4 Overall System Performance

From the analysis of the Table 4.10, the optimal combination for the system is using Faster R-CNN for vehicle detection paired with YOLO segmentation for license plate

detection with an average CER of 0.53. This combination gets the best results in the OCR phase, which is the main goal of the system. The superior performance of both Faster R-CNN and YOLO segmentation significantly contributes to this outcome. Additionally, the segmentation model allows us to deskew the license plate images, enhancing the accuracy of the OCR process.

Stage 1 Model	**Stage 2 Detection**	**Stage 2 Segmentation**
Faster R-CNN	0.58	0.53
YOLOv8	0.63	0.57
SSD	0.64	0.59

Table 4.10: Averge CER of each model from stage 1 with the two models from stage 2.

5 Conclusion

In this thesis, a comprehensive approach to the development of a License Plate Detection and Recognition System based on Deep Learning was presented. The project was structured into three key stages: vehicle detection and classification, license plate detection, and character recognition. State-of-the-art techniques and models were employed in each stage, providing an efficient pipeline for accurately detecting and recognizing license plates.

In the first stage, three different models—Faster R-CNN, SSD, and YOLOv8—were implemented for vehicle detection and classification. Experiments demonstrated that YOLOv8 offered superior performance in terms of speed and accuracy, effectively classifying vehicles into categories such as cars, buses, trucks, and motorcycles.

The second stage focused on license plate detection, where a pretrained YOLOv8 model was utilized. Significant improvements in detecting license plates from various angles and under diverse conditions were achieved, showcasing the model's robustness.

Finally, for character recognition, Tesseract was adopted as the OCR solution. Although initial results were not optimal, methods for preprocessing and enhancing the input data were explored to improve recognition accuracy. Additionally, a dataset from Platesmania was built, which holds potential for training a more specialized OCR model.

Overall, the results demonstrate the effectiveness of integrating deep learning techniques in automatic license plate recognition systems. This work contributes to the growing body of research in the field and offers insights into practical applications such as traffic monitoring, toll collection, and law enforcement.

While this thesis has established a solid foundation for an effective License Plate Detection and Recognition System, several areas for future exploration remain. One key direction is the enhancement of character recognition. Future research should focus on developing a custom-trained OCR model using the Platesmania dataset to

improve character recognition accuracy through targeted training. This would allow for better recognition and interpretation of a wide range of license plate formats and fonts.

The implementation of real-time processing capabilities is another important aspect. Developing a system capable of detecting and recognizing license plates in real-time would significantly enhance its applicability, particularly in traffic monitoring and enforcement scenarios. This may involve optimizing the models for speed and creating an efficient pipeline capable of processing video feeds seamlessly.

The classification of license plates presents another significant opportunity for future research. Algorithms could be developed to categorize license plates based on various attributes, such as region, type (commercial, private, government), and even the issuing authority. Enhancing classification capabilities would improve the overall efficiency of the recognition system, allowing for tailored processing based on specific requirements or regulations associated with different license plate types.

Expanding character recognition capabilities to accommodate different regions and languages would also enhance the system's versatility. Training on a broader dataset that includes various formats and scripts is essential to achieving this goal, allowing for effective recognition of license plates from different countries and regions.

Bibliography

[1] L. Luo, H. Sun, W. Zhou, and L. Luo. "An efficient method of license plate location". In: *Proc. 1st Int. Conf. Inf. Sci. Eng.* 2009, pp. 770–773 (cit. on p. 3).

[2] Z. Sanyuan, Z. Mingli, and Y. Xiuzi. "Car plate character extraction under complicated environment". In: *Proc. IEEE Int. Conf. Syst., Man Cybern.* Vol. 5. 2004, pp. 4722–4726 (cit. on p. 3).

[3] 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. 1998, pp. 1790–1794 (cit. on pp. 3, 4, 8).

[4] 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 (cit. on pp. 3, 9).

[5] G. Heo, M. Kim, I. Jung, D.-R. Lee, and I.-S. Oh. "Extraction of car license plate regions using line grouping and edge density methods". In: *Proc. Int. Symp. Inf. Technol. Converg. (ISITC)*. 2007, pp. 37–42 (cit. on p. 4).

[6] W. Jia, H. Zhang, X. He, and Q. Wu. "Gaussian weighted histogram intersection for license plate classification". In: *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*. Vol. 3. 2006, pp. 574–577 (cit. on p. 4).

[7] W. Jia, H. Zhang, X. He, and M. Piccardi. "Mean shift for accurate license plate localization". In: *Proc. IEEE Intell. Transp. Syst.* 2005, pp. 566–571 (cit. on p. 4).

[8] H. Caner, H. S. Gecim, and A. Z. Alkar. "Efficient embedded neural network-based license plate recognition system". In: *IEEE Trans. Veh. Technol.* 57.5 (2008), pp. 2675–2683 (cit. on p. 5).

[9] C.-T. Hsieh, Y.-S. Juan, and K.-M. Hung. "Multiple license plate detection for complex background". In: *Proc. 19th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*. 1–2 vols. 2005, pp. 389–392 (cit. on p. 5).

[10] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafas. "A license plate-recognition algorithm for intelligent transportation system applications". In: *IEEE Trans. Intell. Transp. Syst.* 7.3 (2006), pp. 377–392 (cit. on p. 5).

[11] J. Matas and K. Zimmermann. "Unconstrained licence plate and text localization and recognition". In: *Proc. IEEE Intell. Transp. Syst.* 2005, pp. 225–230 (cit. on p. 5).

[12] S. Draghici. "A neural network based artificial vision system for licence plate recognition". In: *Int. J. Neural Syst.* 8.1 (1997), pp. 113–126 (cit. on p. 6).

[13] Q. Wu, H. Zhang, W. Jia, X. He, J. Yang, and T. Hintz. "Car plate detection using cascaded tree-style learner based on hybrid object features". In: *Proc. IEEE Int. Conf. Video Signal Based Surveill.* 2006, p. 15 (cit. on p. 6).

[14] K. Kim, K. Jung, and J. Kim. "Color texture-based object detection: An application to license plate localization". In: *Proc. Int. Workshop Support Vector Machines*. Vol. 2388. 2002, pp. 293–309 (cit. on p. 6).

[15] Z. Selmi, M. B. Halima, and A. M. Alimi. "Deep learning system for automatic license plate detection and recognition". In: *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*. 2017, pp. 1132–1138 (cit. on p. 6).

[16] L. Zou, M. Zhao, Z. Gao, M. Cao, H. Jia, and M. Pei. "License plate detection with shallow and deep CNNs in complex environments". In: *Complexity* 2018 (2018), pp. 1–6 (cit. on pp. 6, 10).

[17] 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)*. 2018, pp. 1–10 (cit. on pp. 7, 8).

[18] L. Xie, T. Ahmad, L. Jin, Y. Liu, and S. Zhang. "A new CNN-based method for multi-directional car license plate detection". In: *IEEE Trans. Intell. Transp. Syst.* 19.2 (2018), pp. 507–517 (cit. on p. 7).

[19] T. Nukano, M. Fukumi, and M. Khalid. "Vehicle license plate character recognition by neural networks". In: *Proc. Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS)*. 2004, pp. 771–775 (cit. on p. 7).

[20] S. Du, M. Ibrahim, M. Shehata, and W. Badawy. "Automatic license plate recognition (ALPR): A state-of-the-art review". In: *IEEE Trans. Circuits Syst. Video Technol.* 23.2 (2013), pp. 311–325 (cit. on p. 7).

[21] I. Paliy, V. Turchenko, V. Koval, A. Sachenko, and G. Markowsky. "Approach to recognition of license plate numbers using neural networks". In: *Proc. IEEE Int. Joint Conf. Neural Netw.* Vol. 4. 2004, pp. 2965–2970 (cit. on p. 8).

[22] P. Hu, Y. Zhao, Z. Yang, and J. Wang. "Recognition of gray character using Gabor filters". In: *Proc. 5th Int. Conf. Inf. Fusion (FUSION)*. Vol. 1. 2002, pp. 419–424 (cit. on p. 9).

[23] S. N. H. S. Abdullah, M. Khalid, R. Yusof, and K. Omar. "License plate recognition using multi-cluster and multilayer neural networks". In: *Proc. 2nd Int. Conf. Inf. Commun. Technol.* Vol. 1. 2006, pp. 1818–1823 (cit. on p. 9).

[24] H. A. Hegt, R. J. de la Haye, and N. A. Khan. "A high performance license plate recognition system". In: *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*. 1998, pp. 4357–4362 (cit. on p. 9).

[25] R. Laroca, A. L. Zanlorensi, R. G. Gonçalves, E. Todt, W. R. Schwartz, and D. Menotti. "An efficient and layout-independent automatic license plate recognition system based on the YOLO detector". In: (2019). arXiv:1909.01754. [Online]. Available: https://arxiv.org/abs/1909.01754 (cit. on p. 9).

[26] H. Li, P. Wang, and C. Shen. "Toward end-to-end car license plate detection and recognition with deep neural networks". In: *IEEE Trans. Intell. Transp. Syst.* 20.3 (2019), pp. 1126–1136 (cit. on p. 10).

[27] Z. Xu, W. Yang, A. Meng, N. Lu, H. Huang, C. Ying, and L. Huang. "Towards end-to-end license plate detection and recognition: A large dataset and baseline". In: *Proc. Eur. Conf. Comput. Vis.* Vol. 11217. Lecture Notes in Computer Science. 2018, pp. 261–277 (cit. on p. 10).

[28] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Batra, A. K. C., and C. L. Zitnick. "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755. URL: https://arxiv.org/abs/1405.0312 (cit. on pp. 11, 21).

[29] R. J. L. C. de Nardin, E. P. de Oliveira, M. D. de Oliveira, R. J. P. de Almeida, and G. L. D. L. e Silva. "Open Images: A Large-Scale Dataset for Object Detection and Visual Relationship Detection". In: *International Conference on Multimedia (ICCV)*. 2018. URL: https://storage.googleapis.com/openimages/web/index.html (cit. on pp. 11, 22).

[30] S. Ren, K. He, R. B. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2015, pp. 91–99. URL: https://arxiv.org/abs/1506.01497 (cit. on p. 16).

[31] T. OCR. *Tesseract Open Source OCR Engine*. 2024. URL: https://github.com/tesseract-ocr/tesseract (cit. on p. 20).