PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF MOHAMED BOUDIAF- M'SILA

FACULTY OF TECHNOLOGY

ELECTRONICS DEPARTMENT

N°: ……

DOMAIN: SCIENCE AND TECHNOLOGY

SECTOR: ELECTRONICS

OPTION: EMBEDDED SYSTEMS

Automated Vehicle Detection and Real-time Number Plate Recognition

Dissertation Submitted to the Department of Electronics in Partial Fulfilment of the Requirements for the Degree of Master

Submitted by

Mr. Talal Yassine Nouibat

Supervised by

Mr. Abdelouahab Djoubair Benhamadouche

Presented on: June 25th, 2024, in front of the jury composed of:

Mr. Brik Mourad	University of M'sila	Chairperson
Mr. Benhamadouche Abdelouahab	University of M'sila	Supervisor
Mr. Abed Ahcene	University of M'sila	Examiner

Academic year 2023/2024

Acknowledgements

We thank ALLAH for giving us the health and courage to be able to complete this project. This work is the culmination of a long journey during which we have benefited from the guidance, encouragement, and support of several people, to whom we would like to say a deep and sincere thank you.

First of all, we express our great gratitude to my supervisor Dr. Abdelouahab Benhamadouche, for agreeing to be my supervisor and proposing this subject, for having been present with his advice, his help throughout this journey, and for the patience with which he had demonstrated.

Secondly, i would like to thank my brother Dr. Taha Housseyn Nouibat for his guiding. This work would not be as rich and would not have come to fruition without his help. I thank him for his exceptional technical supervision, patience, rigor, and availability during the preparation of this project.

Through these few words, we extend our warmest thanks to my dear parents and family for the support and encouragement they have always given us.

We would like to thank the Incubator of M'sila, for their guidance and support. Thank you for the educational workshops.

We would also like to thank the members of the jury for having done us the honor of judging and evaluating our work.

Dedication

First of all, I thank Almighty God for giving me the courage, will, and patience to carry out this work despite all the difficulties I encountered.

*To my beloved mother **Saida**, who endured countless sleepless nights and weary days, sacrificing her own comfort for mine since childhood, your unwavering love and support have shaped me into who I am today.*

*To my dear father **Ali**, for all the sacrifices you made and the hardships you endured, your unwavering support and guidance have been a constant source of inspiration for me.*

*To my siblings, **Zwawi**, **Taha**, **Chaima**, and **Ara Maissone**, for being a refugee and a shelter from life's worries and problems, you have been the best siblings one could ever ask for, always there with unwavering support and love.*

*to my entire family, **Nouibat** and **Bachiri**;*

To all my friends, for your unwavering support and delightful company, thank you for being there for me.

To all the teachers in the Department of Electronics Engineering.

Abstract

In this project, we focused our work on the Algerian market and developed a system for automatic license plate recognition (ANPR) in real time. The system was trained using deep learning algorithms on a dataset of Algerian license plates with different photographing angles.

It integrated neural processing units (NPU) to improve image processing performance and recognition speed. We compared the performance of different AI models, including Faster RCNN and YOLO. It showed that YOLOv5 model achieved a high detection accuracy of 99% in real time.

This embedded system can be used in various areas of the Algerian market, such as parking control, vehicle tracking and access point control.

Keywords: Automatic Number Plate Recognition, Neural Processing Unit, Real-Time, Algerian License Plates, ANPR, LPR, ALPR, NPU, Faster RCNN, YOLO.

ملخص

في هذا المشروع، قمنا بتطوير نظام للتعرف التلقائي على لوحات ترقيم السيارات في الوقت الفعلي، مع التركيز على السوق الجزائرية. حيث تم تدريب النظام باستخدام خوارزميات التعلم العميق على مجموعة بيانات لوحات ترقيم جزائرية بزوايا تصوير مختلفة.

تم دمج وحدة المعالجة العصبية للحصول على أداء أفضل في معالجة الصور وسرعة التعرف. قمنا بمقارنة أداء نماذج مختلفة للذكاء الاصطناعي مثل **Faster RCNN** و **YOLO**. أظهرت النتائج أن نموذج **YOLOv5** حقق دقة تعرف عالية بنسبة 99% في الوقت الفعلي.

هذا النظام المتكامل يمكن تطبيقه في مجالات متنوعة مثل التحكم في مواقف السيارات، تتبع المركبات، والتحكم في نقاط الوصول - وذلك كله في السوق الجزائرية.

الكلمات الرئيسية: التعرف التلقائي على لوحة الأرقام، وحدة المعالجة العصبية، الوقت الفعلي، لوحات ترخيص جزائرية.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ALPR	Automatic License Plate Recognition
ANPR	Automatic Number Plate Recognition
CNN	Convolutional Neural Networks
COCO	Common Objects In Context
CPU	Central Processing Unit
DCNN	Deep Convolutional Neural Networks
FPS	Frames Per Second
FRCNN	Faster Region-Based Convolutional Neural Networks
GIoU	Generalized Intersection over Union
LPR	License Plate Recognition
mAP	mean Average Precision
NPU	Neural Processing Unit
OCR	Optical Character Recognition
OpenCV	Open Computer Vision
RCNN	Region-Based Convolutional Neural Networks
ROI	Region of Interest
SSD	Single Shot Multi-Box Detector
VRM	Vehicle Registration Markings
YOLO	You Only Look Once

General Introduction

Currently, automobiles are employed extensively in various facets of human life and business areas as the primary means of transportation in modern societies. This massive expansion has created a very complex set of problems for the management of this huge mass of vehicles, as well as for traffic safety. In addition, traffic infractions, including speeding and running red lights, have become more and more common in public traffic with the increase in the number of newly registered cars.

License plates act as unique identifiers for automobiles, supplying vital information for vehicle ownership. Consequently, vehicle photos have become important resources for person and vehicle identification. Thus, the automated collection and processing of license plate information from digital photographs emerge as useful ways to control public transit.

Automatic Number Plate Recognition (ANPR) stands as a critical component of intelligent transportation systems, exploiting license plate data for vehicle identification. It can be used in various applications, such as highway tolling, community parking management, urban road monitoring, vehicle inspections, traffic statistics, and safety management. The uses of ANPR extend to traffic flow analysis and law enforcement.

In this work, we aimed to create an efficient parking management system and investigate how deep learning models can perform in the ANPR domain. The main goal is to evaluate how these models work at reliably identifying and detecting license plates from pictures and videos taken in various lighting and environmental settings. We tried to improve the accuracy and efficiency of ANPR systems using deep learning methods.

In the first chapter, we provide an introduction and description of the Automatic Number Plate Recognition (ANPR) system. This chapter also covers a comprehensive introduction to image processing.

In the second chapter, we provide a comprehensive overview of the ANPR system development and evaluation methodology. This covers the entire process, from data collection to the implementation and analysis of object detection models, including data acquisition, pre-processing, theoretical foundations, architecture, and performance of Faster R-CNN and YOLO.

In the third chapter, we examine the key performance metrics of object detection models for ANPR: accuracy and inference speed. we assess the performance of three OCR models on Algerian license plate examples, highlighting their strengths and limitations.

Finally, the fourth chapter delves into the practical implementation of the ANPR system, outlining hardware and software components. Key focus is placed on the Orange Pi hardware and how it is integrated into the system. The chapter then examines the system implementation, emphasizing the user interface development and the operational process. It provides a thorough overview of the implementation procedure, highlighting the challenges encountered and the methods used to produce a working ANPR system.

Chapter 1: Background and Overview

1.1 Introduction

This chapter presents a complete introduction and description of the Automatic Number Plate Recognition (ANPR) system. The chapter begins by describing ANPR, tracking its development, and outlining the many ANPR methodologies and their associated constraints.

This chapter also covers a comprehensive introduction to image processing, including the definition of an image, the several types of pictures, image coding, and digital image processing techniques. The importance of optical character recognition (OCR) in ANPR systems is also demonstrated, highlighting its applications and processing methodologies.

1.2 Automatic Number Plate Recognition

1.2.1 What is ANPR?

Automatic Number Plate Recognition (ANPR) is a technology more and more deployed for surveillance. ANPR employs a network of mobile and stationary roadside sensors to gather and analyze vehicle license plates. These sensors automatically cross-reference the acquired data against information stored in the Police National Computer (PNC) and possibly other related systems. The major aim of ANPR is to detect automobiles involved in criminal activities or those in breach of legal restrictions [1].

In basic words, ANPR systems operate by "reading" vehicle registration markings (VRMs), often known as number plates, from digital pictures collected by specialized cameras. The processing capability of ANPR systems varies depending on the specs of the equipment, with some systems capable of processing up to 3,600 license plates every hour [1,2]. Usually, the collected photographs comprise two unique components: a close-up of the license plate itself, referred to as the "plate patch", and a wider-angle view containing the whole vehicle, known as the "overview image". The plate patch picture is then processed by optical character recognition (OCR) technology, which turns the visual data into a text format displaying the license plate number. This collected text is then recorded in an ANPR database along with an electronic log of vehicle movements [3]. This log records the date, time, and position of each capture, giving a full record of vehicle activity. Notably, although the basic function of the system focuses on reading license plates, the software linked with ANPR enables searches against multiple databases using the acquired data. This allows operators to get a response within seconds, delivering significant real-time information when a match is detected. The uses of ANPR technology extend beyond

law enforcement and involve a broad variety of situations, including parking management, tolling, border control, and private security. It is possibly boosting security measures and streamlining many operational operations [2,4].

1.2.2 ANPR Technology limitations

The underlying concepts of ANPR are fairly simple but difficult to implement. Capturing a sufficiently good image of the plate is the most challenging aspect of ANPR, regardless of light conditions, vehicle speed, and plate condition.

It is misleading to infer that ANPR systems can all be used interchangeably and will produce the same quality data. Some systems work with sub-par cameras or low-quality images to produce a less reliable read. High-grade ANPR cameras have infrared and other factors that enable them to capture good images and read poor photos in low-light situations or at night.

In ideal conditions provided by high-quality modern systems, the overall read rates for ANPR are assumed to be between 90% and 94%. However, other facts could contribute to high levels of misreads, such as dirt, cloned, or fake number plates. Alternatively, the older ANPR systems have been regarded as highly unreliable, with performance rates between 60% and 80%.

1.3 License plate recognition process

Before getting into the ANPR system, it is essential to understand the fundamental functionalities that are involved. One way to define the role of the ANPR system is through a two-stage process that works collectively to recognize the license plates. Figure 1.1 shows a car with license plate.

Figure 1.1 Vehicle with European license plate

- The first phase's task is to localize the license plates. Therefore, during this step, the original image is examined to determine the area in which the car's license plate is located.

- After the analysis and identification of the desired region, the ANPR system switches to the second stage, which is character extraction. This step now turns attention towards isolating the characters or letters in the license plate number.

An initial stage of the 2-stage process is completed, as subsequent operations are reliant on the extracted characters, which must be used as the basis for subsequent processing. The extracted characters based on the 2-step procedure will be passed into the next stages of the system, where they engage in various operations that compare with the state's database. The following section will delve deeper into this to understand the finer details of the complete workflow [5].

1.3.1 Locating the license plate

The fundamental process for finding license plates is at the heart of ANPR technology. This algorithm carefully examines the recorded picture (usually in black and white) to identify the license plate in the scene. The white pixels should ideally accurately depict the plate; however, car logos or reflections might complicate things. As a result, the system makes use of predefined criteria, including the expected size of a license plate, to distinguish it from the surrounding clutter [5] (Figure 1.2). For the next phase, which involves extracting the single letters written on the plate, this precise localization is essential.

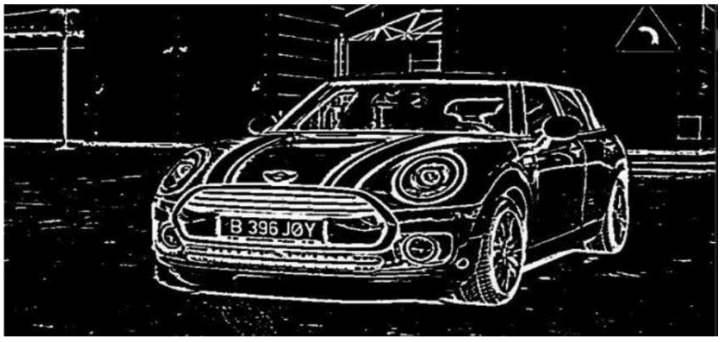


Figure 1.2 Extraction of the location

1.3.2 The license plate features extraction

When the license plate location algorithm has already outlined the general area in which the license plate might be placed in the captured image, the next step is to define the rectangle that holds the license plate. Essentially, at this phase of the algorithm, the image can be viewed as a grid, similar to a chessboard. To properly allocate the plate, the algorithm identifies the top and the bottom edges of the rectangle shown by their y-coordinates. Overall, this step can be defined as the setting of the plate's height in the image.

Then, the second operation is to draw the top and bottom edges using x-coordinates, with a focus on the horizontal positions. Thus, considering all four corner coordinates, the algorithm delimitates a specific space bound within a rectangle that encompasses the license plate. Such a boundary is critical for the following step, as individual characters of the license plate are later cut out and identified for the final implementation (Figure 1.3) [5].

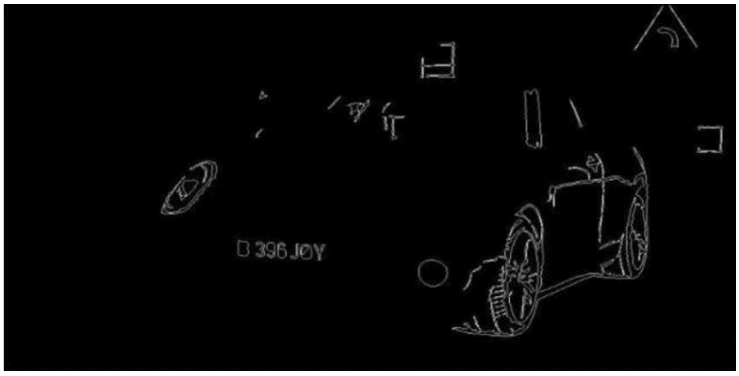


Figure 1.3 Vector of the vector matching chars.

1.4 Applications of License Plate Recognition System

All cars must have a license plate, which serves as a crucial means of identification for the driver. License plate recognition technology simplifies getting this information. The following list of application examples presents a summary of several areas where license plate recognition technology is implemented regularly.

- **Parking Management:** A license plate recognition system may be put at the entrance of the parking structure to monitor passing automobiles. When a car reaches an input gate, its number plate is immediately identified and recorded in the database. The number plate is scanned again upon the departure of the automobile, and the difference in time is used to determine the parking cost. The parking record data could specify the current parking spots available [6].

- **Access Control and Security:** License plate recognition is used in various firms and facilities to provide access only to cars owned by authorized staff. The license plate recognition technology placed at the entry gate can automatically open the gate for returning residents by recognizing their cars stopped at the threshold. Compared with the previous technique, where the corporation normally hires security guards or puts specific gadgets on the cars for identification, the license plate recognition system gives a more efficient and flexible manner of access control [7].

- **Speed/Law Enforcement:** Many cities have placed traffic cameras to monitor the movement and flow of cars across the road network. By employing license plate identification on these footages, it is possible to monitor the duration of time for a specified vehicle transition between two sites. Since the distance between the two license plate recognition sites is static, the vehicle's average speed may be determined based on the gotten photos. Automated tickets may be issued and delivered to the owner of the car by cross-referencing the license plate recognition information with an existing database. It may also allow law enforcement agencies to monitor and detect suspicious cars and unlawful usage of bus lanes in real time [8].

- **Tracking and Traffic Management:** In the instance of a stolen vehicle, or a car witnessed at the scene of a crime, license plate recognition devices spread in an area could monitor the subject cars. Each license plate recognition system has a known location and may tell the precise moment a certain vehicle is detected. From this data, a tracking map may be generated to determine the subject vehicle's whereabouts. Similarly, license plate recognition technology may also produce a mapping of traffic flow and be used as a vehicle counting mechanism, assisting in the compilation of traffic management data [9].

- **Customs/Immigration:** At border crossings and customs inspection locations, a considerable volume of vehicle traffic typically occurs. Using license plate recognition technology allows for the automatic scanning and database comparison of license plates for all vehicles that are nearing the border. The database might contain stolen automobiles, linked criminal records, and other vehicles of interest. If anything is detected in the database, the vehicle might be sent to a different region for additional inspection [9,10].

- **Mobile Vehicle Readers**: By placing a license plate recognition system in a police vehicle, additional options are accessible to law enforcement officials. License plates may be recognized from a distance without disturbing the driver. The police may verify the recorded license plate picture against a database of stolen or doubtful automobiles. If there is a positive match, the system may inform the police for further action [10].

1.5 Algerian License plate

Every country has its legislation specifying the form and content of registration plates. In Algeria, the License plate number contains 3 groups of numbers separated by a space (Figure 1.4):

- **1st group:** 5 digits, maximum 6 digits, which correspond to the serial number of the vehicle.
- **2nd group:** composed of 3 digits: the first digit identifies the type of vehicle (1 conventional vehicle, 2 for a truck, 3 commercial cars, 4 buses, 5 semi-trailer tractors, 6 tractors, 7 machines, 8 trailers, 9 motorcycles), the next two digits refer to the year of the vehicle first circulation.
- **3rd group:** 2 digits that identify Algerian Wilaya number (Provinces ID From 01 to 58 currently).

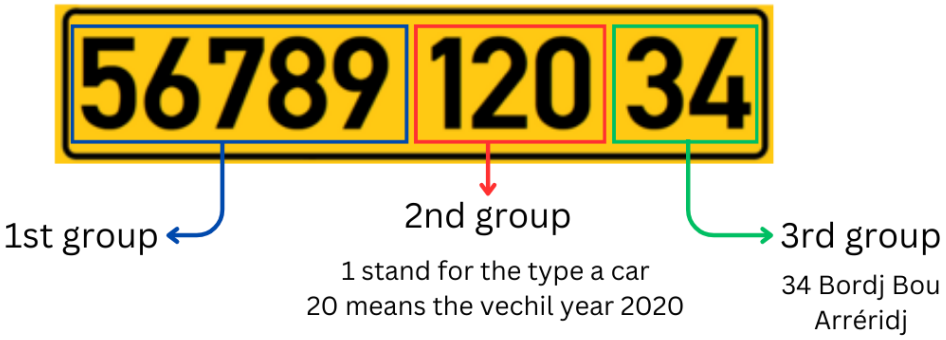


Figure 1.4 Algerian License plate.

1.6 Image processing

1.6.1 Definition of Image

The term 'image' has various meanings. It can also be referred to as a visual depiction of an object, such as a photograph or painting. In addition, it can refer to a mental representation of an object, like a dream or hallucination. The term (image) will be used in this thesis to describe a digital representation of an object.

1.6.2 Image Acquisition

The process of converting a physical object picture into a digital image is known as image acquisition, and this can be accomplished using various devices, such as cameras, scanners, and webcams. The digital camera is the most common type of device for capturing images. Light is captured by digital cameras and converted into an electrical signal, which is subsequently processed and saved as an image file [2]. This process is illustrated in Figure 1.5.

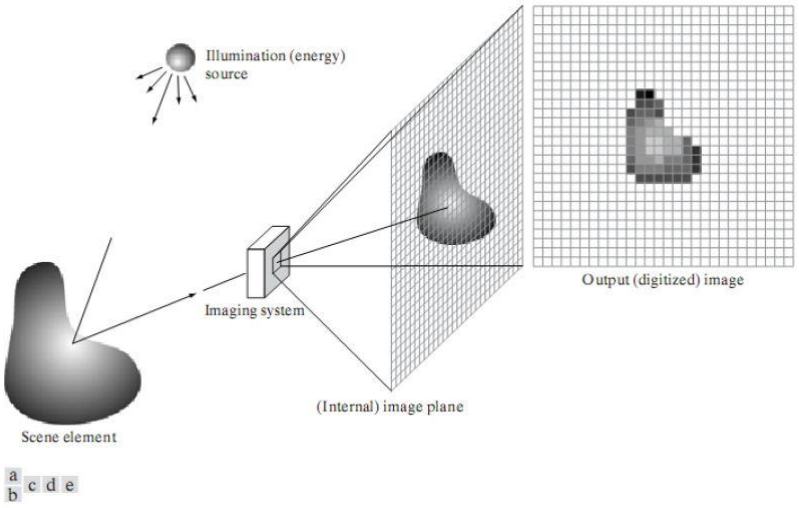


Figure 1.5 digital image Acquisition Process.

1.6.3 Image Processing

Image processing is the course of transforming a digital image. This can be done for a variety of reasons, such as improving image quality, removing noise, or extracting features. Different image processing techniques exist, and the specific ones used will depend on the desired outcome [3].

1.6.4 Image Storage

Image storage is the course of saving a digital image to a computer or other storage device. The specific image format used will depend on the application because of the many available formats. Some common image formats include JPEG, PNG, and GIF [4].

1.6.5 Image types

There are two primary types of digital images are raster images and vector images (Figure 1.6).

- Raster Images are composed of a grid of pixels, with each pixel representing a small square of color. The size and quality of a raster image are determined by its resolution. The quality of the image increases with the number of pixels in it.
- Vector images are composed of mathematical equations that define the shapes of the objects in the image. Unlike raster images, vector images can be scaled to any size without losing quality [1].

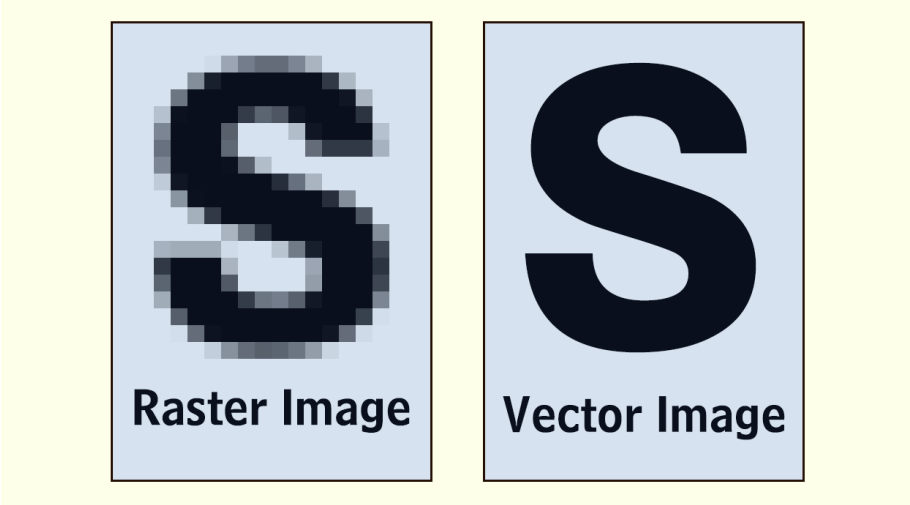


Figure 1.6 Differences between Raster image & vector image.

1.6.6 Image Coding

The pixel is the primary tool for determining the definition of a raster digital image. An image depends on the color coding of the pixels; therefore, it can be:

1.6.6.1 Binary Image (Black and White)

In this image, a pixel can be represented by two different values: 0 for black and 1 for white shown in Figure 1.7.

Figure 1.7 Binary image 0 &1.

1.6.6.2 Gray-scale Image

The Gray-scale in digital images represents the brightness of a pixel. Because a pixel is coded on one byte, it has 256 values, which allow for more nuances than a basic black and white (Figure 1.8).

Figure 1.8 Gray-scale image from 0 to 255.

1.6.6.3 Color Image

There is a range of color-coding modes available on computers. The most widely used color space for image processing is the red, green, and blue (RGB) color space. This space is based on the additive synthesis of colors. The colors are got by mixing the three primary colors: 8 bits are associated with each color, and each "color" pixel is coded on 3 bytes to form 16 million different colors (Figure 1.9).

Figure 1.9 Original image (color image RGB).

1.6.7 Digital Image Processing

Digital image processing is the set of operations performed on an image to improve its readability and facilitate its interpretation. This is the case, for example, for operations such as contrast enhancement, noise removal, and blur correction. It is also the set of operations performed to extract "information" from the image, such as segmentation and contour extraction. Before image processing, preprocessing operations can also be performed, which are all techniques aimed at improving the quality of an image.

1.6.8 Image Filtering

The principle of filtering is to adjust the pixel values of an image, often to improve its appearance. In actual use, this means that the original image's pixel values are used to create a new image. Two forms of filtering can be used in image processing: nonlinear and linear.

1.6.8.1 Linear Filtering

The process of replacing each gray-scale level with a linear combination of the gray-scale levels of adjacent pixels is essentially known as linear filtering. Linear filters are defined by their impulse response (Figure 1.10).

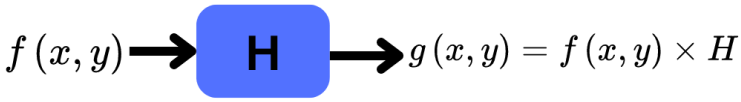


Figure 1.10 Linear system equation.

f: represents the input image.
g: represents the output image
H: represents the mask.

Gaussian Filter: The Gaussian filter is one of the most popular linear filters because it produces high-quality results while remaining simple to set up. This filter eliminates what is known as Gaussian noise. This sort of filter uses the weighted average of the ranges, as shown in Figure 1.11.

Figure 1.11 Gaussian filter using 5x5 mask.

1.6.8.2 Nonlinear Filtering

Nonlinear filtering could improve the drawbacks of linear filters, principally the poor retention of contours. Adjacent pixels intervene in this nonlinear filter based on a nonlinear law (Medium filtering, for instance).

Median Filter: replacing a pixel with the median value of the analysis window cantered on the pixel is known as median filtering. The mask size is determined by the noise variance and the size of the processed image significant details. This filter is extensively used to minimize impulsive

noise in a picture that might be of different causes (dust, small clouds, etc.) and possibly result in little spots whose distribution on the image is random. The benefit of using this filter is that it keeps the image sharpness (Figure 1.12).

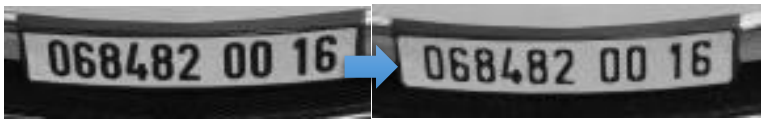


Figure 1.12 Median filter for every 2px.

1.6.8.3 Binarization (Thresholding)

Binarization allows the conversion from a gray-scale image to a binary image made of two gray-scale levels, 0 and 255, which correspond respectively to black and white, which is easier to process. Then, an object in contrast to a background, can be distinguished because of binarization. There are two classes (black and white) for the gray-scale levels; black is used to show the shape and white to represent the background. A reference value called "threshold" is used to compare each pixel intensity in the thresholding (binarization) process. Local and global binarization are the two different forms of binarization.

a) Global Binarization (Average)

Global thresholding is selecting a threshold that is customizable but identical across the entire image. For a 256-gray scale, this value is between 0 and 255, often in the median range. Each pixel is compared to this threshold: in direct polarity, pixels with lower gray-scale levels are assigned to "white" (0) and those with greater levels to "black" (1) shown in Figure 1.13.

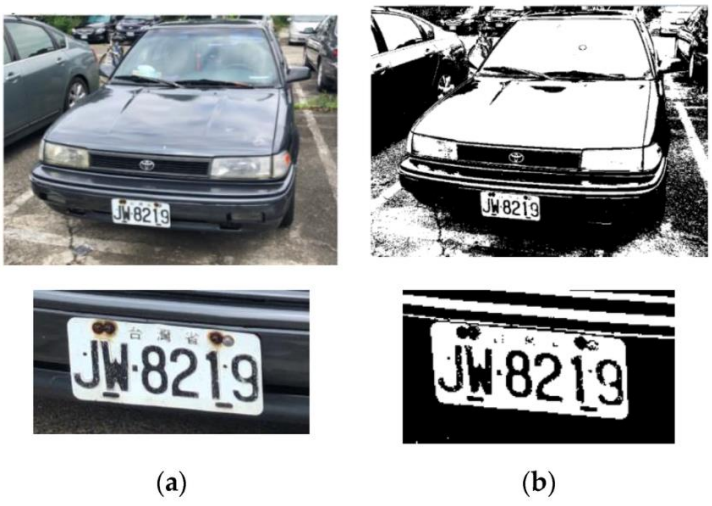


Figure 1.13 Binarization (a. before b. after binarization)

b) Local Binarization

In local binarization, a pixel's assignment is determined not only by the pixel itself but also by its adjacent. Although this method is resilient, it has the disadvantage of being sluggish, since the binarization of each pixel requires the examination of its pixel's adjacent.

$$\mu = \frac{1}{(2n+1)^2} * \sum_{k=-n}^{n} \sum_{l=-n}^{n} f(i+k, j+l)$$
$$\sigma = \sqrt{\frac{1}{(2n+1)^2} * \sum_{k=-n}^{n} \sum_{l=-n}^{n} |f(i+k, j+l) - \mu|^2} \qquad (1.1)$$

$f\ (i, j)$ represents the central pixel for a range of (2n+1) ×(2n+1).

For each pixel in the image, it is compared in direct polarity to the following threshold:

$$S = \mu - k \cdot \sigma \qquad (1.2)$$

μ: the mean of the (2n+1) ×(2n+1) range of the pixel being processed.

σ: the standard deviation of the (2n+1) ×(2n+1) range of the pixel being processed.

k: binarization coefficient (with 0≤k≤1).

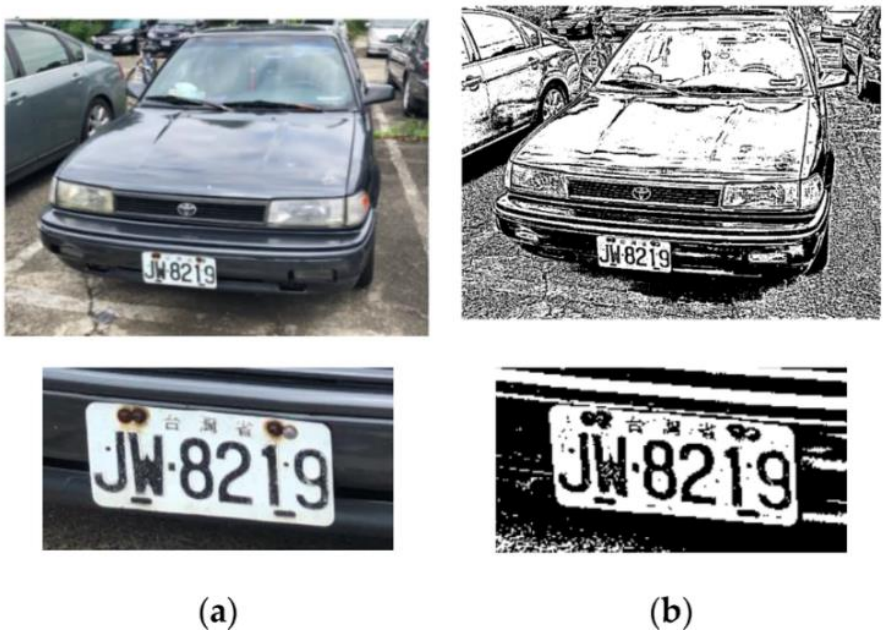


Figure 1.14 Binarization local thresholding, (a) before, (b) after thresholding.

1.6.8.4 Character normalization

Normalization is expanding or contracting all image points. It is an operation that comprises making the image a standard or fixed size. However, if the fixed size is too small, information may be lost; if it is too large, the recognition step will run slowly [12], so good normalization will increase the system's processing speed and significantly reduce the recognition time while retaining the information included in the image. Normalization in our situation comprises taking the same size (40*40) for all the photos that potentially had various sizes due to cropping (Figure 1.15).

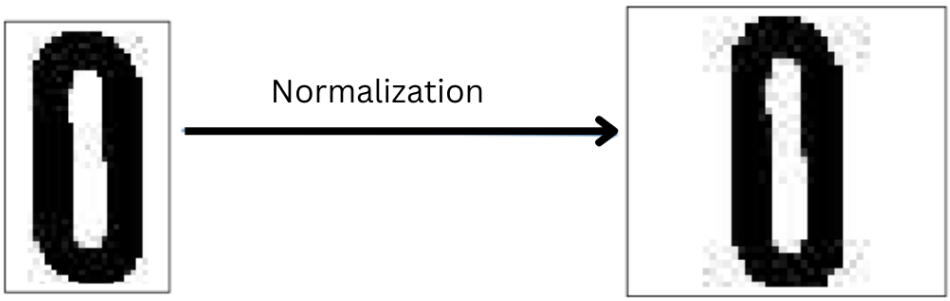


Figure 1.15 normalization character 0 from size 21x45p to 40x40p.

1.7 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is the process by which a computer recognizes and extracts printed or handwritten text characters from digital images. The computer analyzes the text and converts the characters into a code that can be used for data processing, such as a street sign or traffic sign in a picture [11].

OCR engines have been integrated into software programs that specialize in text processing, including receipts, invoices, checks, and legal billing documents. The OCR is useful for:

- Data entry for commercial documents, such as bank statements, receipts, invoices, passports, and checks;
- Automatic identification of license plates and recognition of traffic signs;
- At airports, passport recognition and information extraction;
- Important data is automatically extracted from insurance paperwork;
- Getting contact information from business cards;
- Enabling the search ability of electronic copies of printed materials, such as Google Books;
- Assistive technology for those who are blind or visually impaired;
- Converting scanned documents to PDFs so they can be searchable.

1.8 Conclusion

This chapter provided a complete description of the ANPR system, including its creation, methods, and limits. It also covered the fundamentals of image processing. These techniques are critical components of ANPR systems. Understanding these concepts is important for creating effective and accurate ANPR systems. The chapter also emphasizes the value of ANPR systems in a variety of applications, such as traffic management, crime prevention, and intelligent transportation systems. This chapter's knowledge will lay the groundwork for the design of an efficient ANPR system, which will be covered in the next chapters.

Chapter 2: Methodology

2.1 Introduction

In this chapter, we provide a comprehensive overview of the methodology employed in the development and evaluation of our Automatic Number Plate Recognition (ANPR) system. It is structured to cover the entire process, from data collection to the implementation and analysis of object detection models. This includes the acquisition and pre-processing of data, the theoretical underpinnings of object detection, detailed architecture, and performance metrics of two prominent models: Faster Region-based Convolutional Neural Network (Faster R-CNN) and You Only Look Once (YOLO).

2.2 Data Collection

The field of LPR has made significant improvements in recent years. However, the training data characteristics have a significant performance impact on LPR models. To achieve robust and generalizable model performance, it is crucial to have a well-structured dataset that encompasses diverse image variations (backgrounds, lighting, angles) and high-quality license plate representations. The strategies employed in this section outline how to get a comprehensive dataset that is tailored to the specific requirements of LPR for Algerian vehicles.

2.2.1 Data Sources

Toward getting a variety of Algerian car photos with legible and high-quality license plates, a thorough search was performed on Google Images. A variety of search expressions were used, such as "Algerian cars", "Algerian license plates", and "vehicles with Algerian registration plates". The aim was to encompass a wide range of vehicle types and plate formats that are prevalent in Algeria.

An additional source of data was the well-known Algerian e-commerce platform, Ouedkniss. The website provides a huge library of photographs uploaded by users, many of them containing automobile pictures (Figure 2.1).

Figure 2.1 A sample dataset collected for Algerian vehicles pictures

2.2.2 The manual data collection process

To ensure the quality and validity of the gathered data, a careful manual selection process was conducted. Each photograph was extensively examined based on pre-defined criteria:

- **Relevance:** Pictures have to show cars from Algeria that are obvious via their license plates. Foreign-registered cars and cars with covered or unreadable license plates were not included.
- **Quality:** The quality of the image was important. Priority was given to high-resolution photos with little blurring and a good focus on the license plate. To prevent affecting the performance of the LPR model, low-resolution or fuzzy photos were removed.
- **Diversity:** The selection process deliberately aimed to include a wide range of image variations, such as varied backgrounds (urban, rural, highway), lighting situations (daytime, nighttime), and vehicle orientations (front, back, and side). This focus on diversity sought to strengthen the model's generalizability and ensure its capacity for successfully detecting license plates in diverse real-world circumstances.

2.2.3 Data Acquisition

After the selection procedure, photos were manually downloaded while respecting all copyright limitations and moral principles (explained in the Ethical Considerations section). Manual downloading allowed for more precise control over the selection process, ensuring that the pre-established criteria were followed and that improper or irrelevant content was excluded, even though automated download scripts were available.

2.2.4 Dataset Characteristics

- **Number of images:** By merging information from the Google Images searching engine and Ouedkniss, the painstaking data collection process produced a dataset that included over 1200 images. This sample size is in line with industry standards for training LPR models and offers enough data to produce a strong model.
- **Categories:** The dataset is mostly divided into one category: cars with license plates from Algeria. For future research endeavors, it could be explored to categorize vehicles according to types (cars, trucks, motorcycles).
- **Diversity and quality:** Priority was given to image diversity in terms of angles, lighting, and backdrops in the selection criteria. In addition, during data preprocessing, selective photos underwent manual cleaning and color modifications to improve license plate readability. These measures produced a high-caliber dataset that was appropriate for training LPR models.

2.2.5 Data Preprocessing

The dataset could further be refined and made ready for model training by implementing a data preparation stage after data collection. This stage includes:

- **Cleaning:** A thorough cleaning procedure was used to remove duplicates and unnecessary photos (those that did not have visible license plates).
- **Cropping and color correction:** To enhance the legibility of license plates and make labeling simpler, photos were manually cropped and color-corrected.

2.2.6 Labeling License Plates

The license plates were manually annotated using Roboflow to prepare the photos for training the LPR models.

Robotflow is a platform designed to streamline the process of building, training, and deploying computer vision models. It provides data annotation (labeling), and management tools so users can prepare datasets for machine learning. Additionally, Robotflow provides resources to train models.

Labeling Tool: The labeling procedure was performed using Roboflow. With Roboflow, you can accurately and quickly label items in photos with an advanced annotation tool. RoboFlow's export method offers a versatile solution, enabling the creation of distinct labeling files tailored to

specific computer vision models, in that way facilitating seamless integration and optimization. Also, for labeling images, we use one class (plate) for license plate detection purposes shown in Figure 2.2.

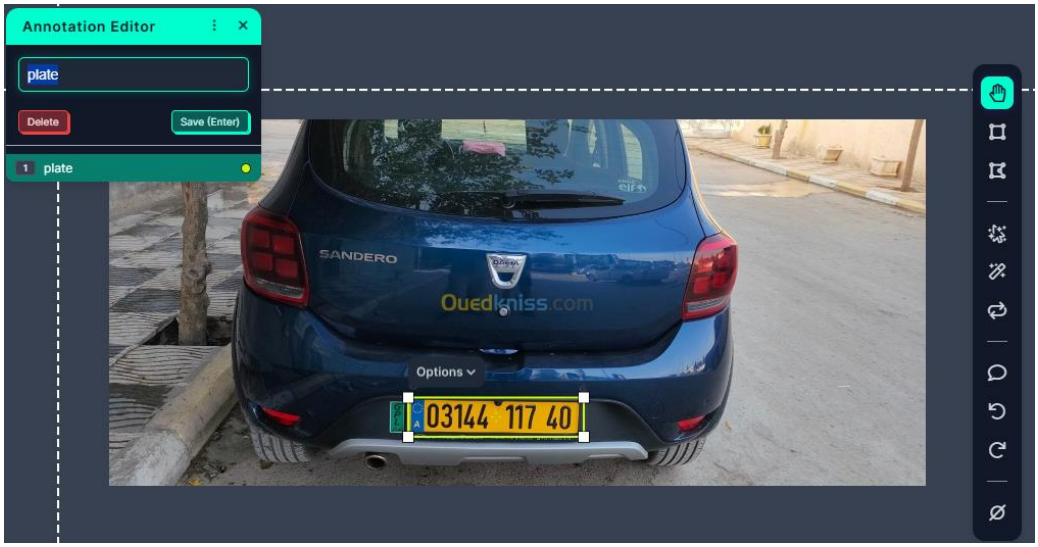


Figure 2.2 Image labeling using binary class (plate)

Labeling Procedure: Bounding boxes were drawn around the license plates in each photograph to provide annotations. To ensure high-quality annotations, it was necessary to note the license plate coordinates.

Labeling Requirements: Only the pertinent region was included in the bounding boxes, which were tightly drawn around the license plates. To guarantee that the training data was of high quality, labeling consistency was kept across the dataset.

2.2.7 Dataset Split

The dataset was distributed into training, validation, and testing sets: 70% training, 15% validation, and 15% testing. This division increases the model's robustness and guarantees that it may be assessed in a variety of settings.

2.3 What is object detection?

One type of computer vision task is object detection, which involves recognizing and locating items in pictures or videos. It is a critical component of many applications, including surveillance, self-driving cars (Figure 2.3), and robotics. Object detection techniques can be classified into two types: single-shot detectors and two-stage detectors (Figure 2.4) [12].

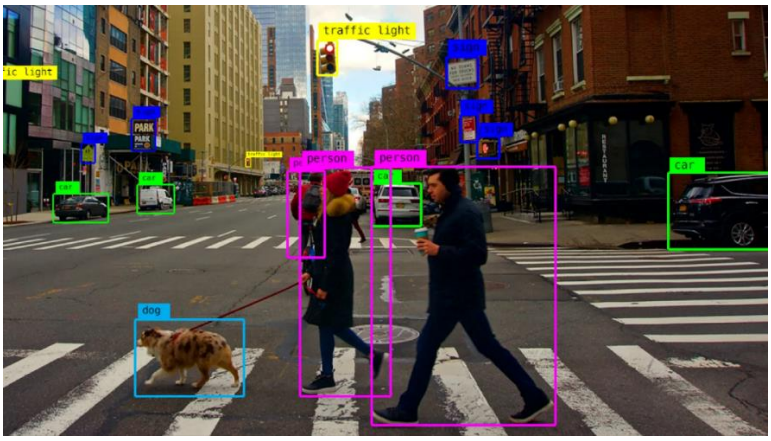


Figure 2.3 Object Detection

In 2014, Ross Girshick and his colleagues at Microsoft Research developed the R-CNN (Regions with CNN Features) model, which was one of the first successful attempts to address the object detection problem using deep learning. This model detects and locates objects in images by combining convolutional neural networks (CNNs) with region proposal methods [13].

There are two types of object detection algorithms, depending on how many times a network passes the same input image.

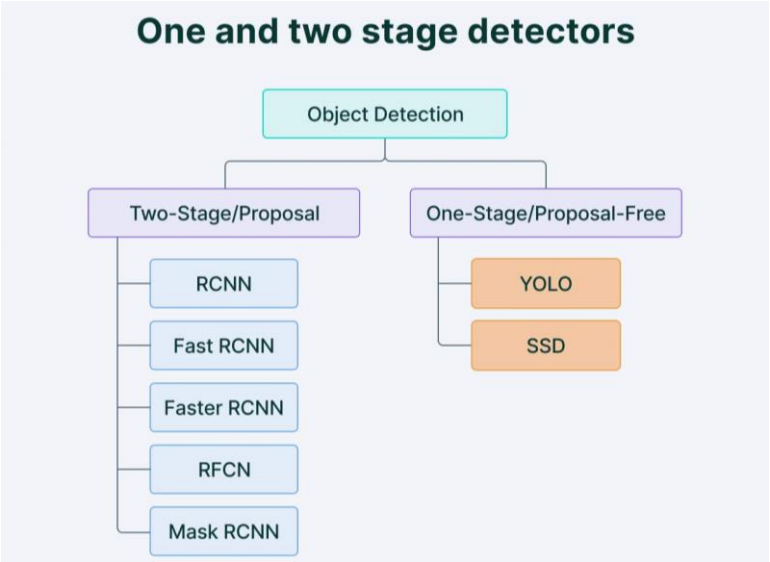


Figure 2.4 Object detection algorithm stages

2.4 Faster Region-based Convolutional Neural Network

Faster Region-based Convolutional Neural Network, or Faster R-CNN, is an approved form of CNN that is commonly used for object detection. It takes an image as input and returns bounding

boxes around objects of interest labeled with their class [14]. We will use it in our work to identify license plates in a picture (Figure 2.5).

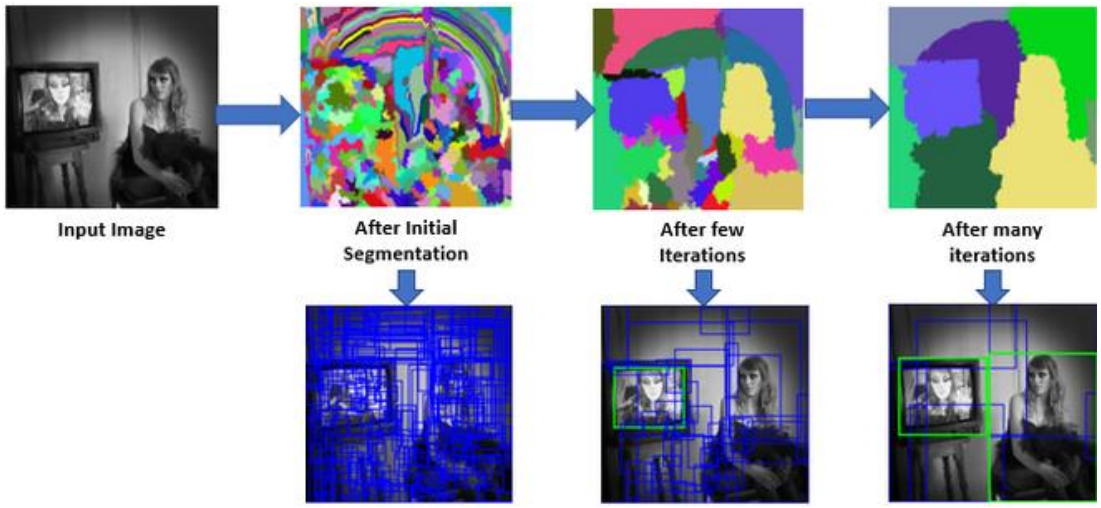


Figure 2.5 Faster R-CNN search selective

Understanding the original R-CNN is necessary before discussing faster R-CNN. The primary concept is employing the search-selective approach to identify regions of interest, which are then passed to the Convolutional Neural Network. R-CNN attempts to identify regions that may contain objects by grouping comparable pixels and textures into boxes [14]. 1300 suggested boxes are used based on search selection. These bounding boxes will be passed to the CNN model that has already been trained, where regression analysis is done to determine the difference between the predicted and ground-truth bounding boxes.

2.4.1 Faster R-CNN architecture

Faster R-CNN is built of many integrated components that together evaluate an input image to yield a list of bounding boxes, each associated with a label and a probability score [15]. The input images are encoded as three-dimensional arrays (height x width x depth), which serve as tensors fed into a pre-trained convolutional neural network (CNN). These tensors flow through the CNN up to an intermediate layer, culminating in the formation of a convolutional feature map [15,16]. Figure 2.6 Illustrate Faster R-CNN architecture.

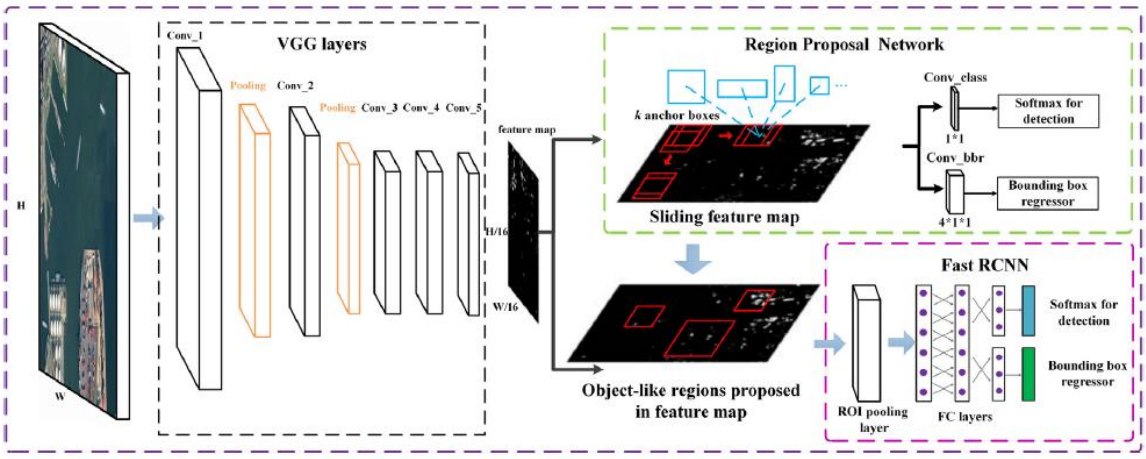


Figure 2.6 Faster R-CNN architecture

Subsequently, a predetermined number of regions or bounding boxes are identified by the Region Proposal Network (RPN) using the convolutional feature map. These bounding boxes are constructed using anchors, which are fixed-sized reference boxes uniformly dispersed across the source image. Each anchor handles two essential questions: whether it contains a meaningful object and how it should be altered to suit the object [16].

Region of Interest (ROI) pooling is used when possible relevant objects have been identified and their locations have been determined. This phase pulls features about the relevant items into a new array, employing both the features produced from the CNN and the bounding boxes. Ultimately, the R-CNN module leverages this array to classify the information within each bounding box and adjust the coordinates to better fit the recognized item [17].

2.4.2 Convolutional Neural Network

The first stage in the faster R-CNN architecture is to use a convolutional neural network (CNN) that has already been trained for classification and feature extraction from an intermediate layer. This phase is crucial to understanding the functioning and effectiveness of the faster R-CNN model. One popular illustration of a pre-trained network of this type is VGG-16 [18](Figure 2.7).

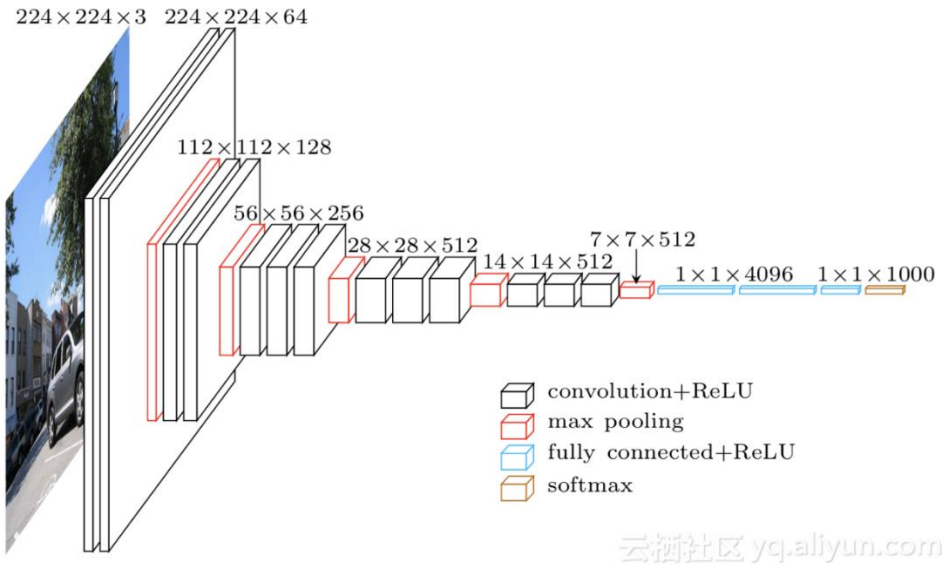


Figure 2.7 VGG-16 architecture (convolutional layers)

The abstractions established by the previous layer are expanded upon by each convolutional layer in VGG-16. To detect more complicated structures, later layers recognize patterns inside the edges that the earlier layers focused on learning, like edges. This approach results in a convolutional feature map that encodes complete information about the image while preserving the spatial arrangement of objects. Because of the pooling techniques used in between convolutional layers, this feature map has substantially more depth, even if its spatial dimensions are smaller than those of the original image. The number of filters that the convolutional layers learn rises with depth [18]. Figure 2.8 illustrate an overview of the convolutional feature maps

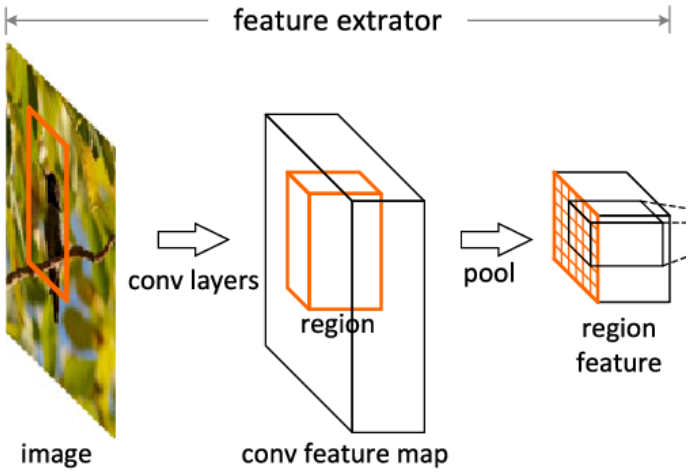


Figure 2.8 Overview of the convolutional feature maps

2.4.3 Region Proposal Network (RPN)

The Region Proposal Network (RPN) is a critical component of the Faster R-CNN design, and it generates high-quality ideas for likely item positions within images. To accomplish this goal, the RPN employs a succession of convolutional layers, beginning with a convolutional layer of 512 channels and a 3x3 kernel size, then two convolutional layers with 1x1 kernels shown in Figure 2.9. The number of channels in these successive layers is determined by the number of anchors at each spatial point on the feature map [15].

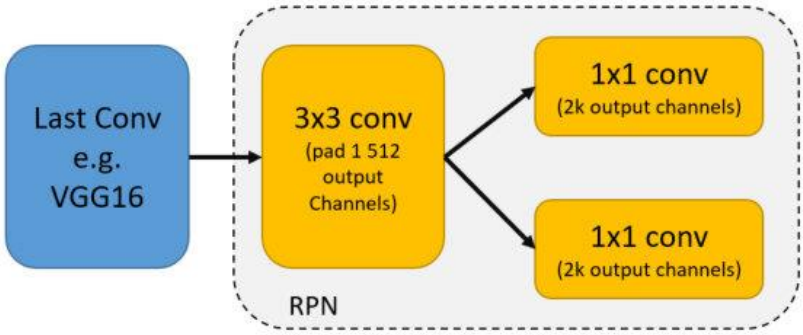


Figure 2.9 Convolutional implementation of an RPN architecture. k corresponds to the number of anchors

The RPN's primary function is to generate object proposals by processing input anchors. Each anchor produces two results: the probability of containing an object and the bounding box regression values, which alter the anchor's position and size. The classification layer makes two predictions per anchor, showing either it comprises an object or not. Concurrently, the regression layer generates four coordinates per anchor, allowing the bounding boxes to better correspond with the actual objects [15,16].

2.4.3.1 Anchors

Anchors are pre-configured bounding boxes that are distributed equally across the image at various sizes and aspect ratios, serving as reference points for anticipating item positions.

A set of anchors is produced for each point in the feature map grid, ensuring that all object placements are covered. For example, in the VGG-16 network, the anchors are 16 pixels apart, leading to the creation of a dense grid of proposals. This configuration enables the RPN to capture objects of varying sizes and shapes effectively. As shown in Figure 2.10, we have 18×25=450 points, Anchors, these 450 points are the center of the bounding box [18].

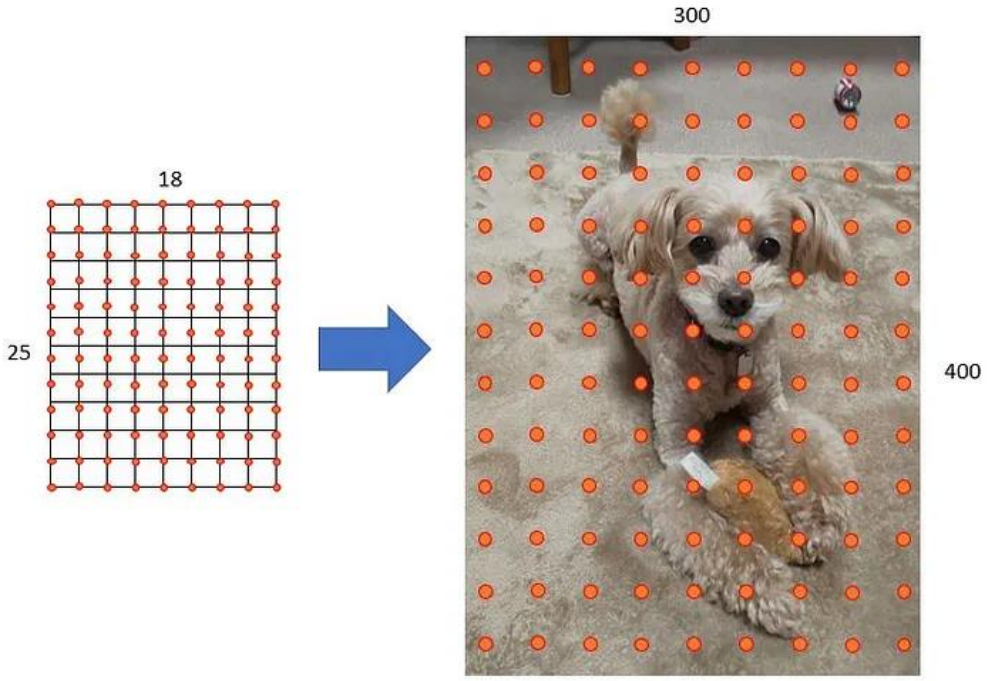


Figure 2.10 Set Anchors in feature maps and apply them to the original image

2.4.3.2 Training Loss Functions

The RPN is trained by optimizing two main components: objectness classification and bounding box regression. The goal is to precisely determine whether each anchor includes an object and refine the anchor's coordinates for precise localization [18].

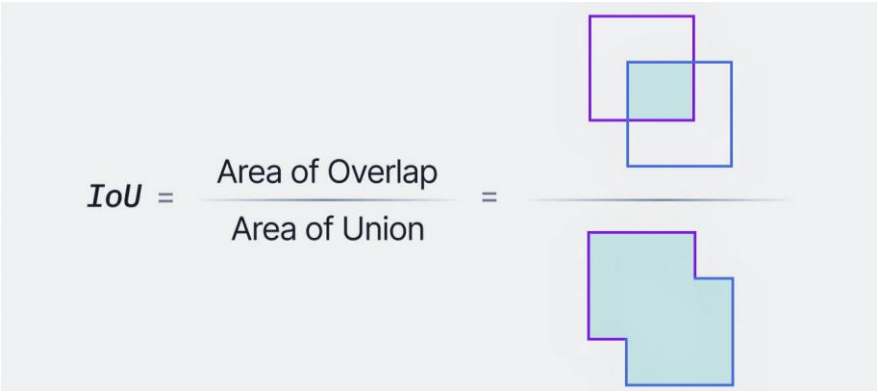


Figure 2.11 IoU metric representation

- **Objectness Classification:** The RPN divides anchors into two types: foreground anchors, which overlap a ground-truth object with an Intersection over Union (IoU) larger than 0.5, and background anchors, which do not overlap any ground-truth object or have an IoU (Figure 2.11) lower than 0.1.

- **Bounding Box Regression:** is a technique used in computer vision tasks, particularly in object detection, to refine the location and size of bounding boxes around objects of interest within an image.

2.4.3.3 Post-Processing

Post-processing is required to enhance the RPN accuracy and quality of the final item detections.

- **Non-Maximum Suppression (NMS):** NMS eliminates duplicate proposals by iterating over the sorted list of proposals (ordered by score) and replacing proposals with an IoU greater than a predetermined threshold (often 0.6) with high-scoring proposals. This stage guarantees that only the most relevant and non-overlapping suggestions are selected.
- **Proposal Selection:** After applying the NMS, the top N proposals are chosen based on their scores. While the original work uses N = 1500, this number can be reduced to as low as 60 without significantly affecting detection effectiveness. This step refines the set of ideas sent to Faster R-CNN's subsequent stages for comprehensive object categorization and localization.

2.4.4 Region of Interest Pooling (RoI Pooling)

RPN generates a set of bounding boxes with no assigned class. As a result, we have divided these object proposals into groups.

To classify each proposal into a defined number of classes, the R-CNN requires fixed-sized feature maps, which may be extracted for each proposal using region of interest pooling (Figure 2.12). This allows for a faster R-CNN to reuse existing convolutional feature maps [15].

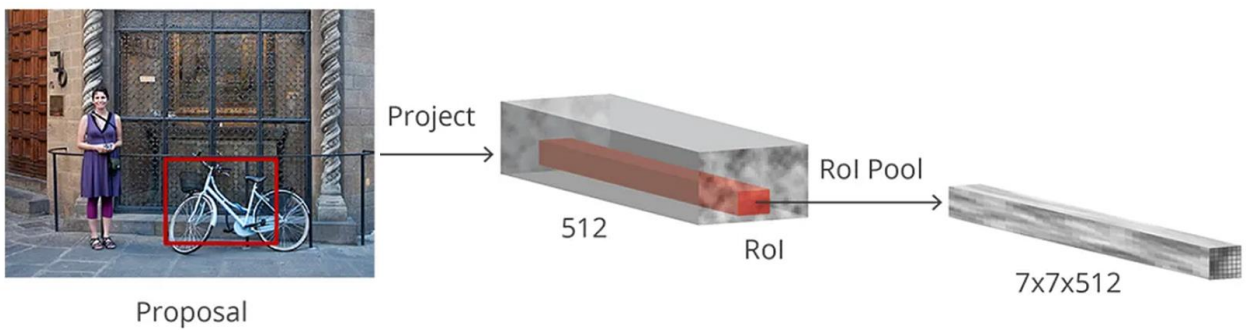


Figure 2.12 RoI pooling

2.4.5 Region-based Convolutional Neural Network (R-CNN)

The last stage of the Faster R-CNN pipeline is the region-based convolutional neural network (R-CNN), which uses the characteristics that were retrieved from the previous layers to refine bounding boxes and classify data. The main goals of the R-CNN in Faster R-CNN are to refine the bounding boxes for these proposals based on the projected class and categorize the suggested regions into one of several predetermined object classes.

R-CNN starts by flattening the relevant feature map for each proposed area. Then, to add non-linearity and enhance the model's capacity to learn complex patterns, this flattened feature map is run through two fully linked layers, each with 4096 units and using Rectified Linear Unit (ReLU) activation functions Figure 2.13. After these layers, the R-CNN uses two unique, fully linked layers for additional processing:

- **Classification Layer:** In this layer, there are N+1 units, where N is the number of object classes and the extra unit is the background class. With this structure, each proposition can be classified by the network as background or as one of the object classes.
- **Bounding Box Regression Layer:** is composed of $4N$ units and intended to produce four regression values (Δcenter_x, Δcenter_y, Δwidth, and Δheight) for each of the four N different classes. By modifying the bounding boxes' coordinates, these values help to better fit the identified objects by adjusting the boxes' positions and sizes.

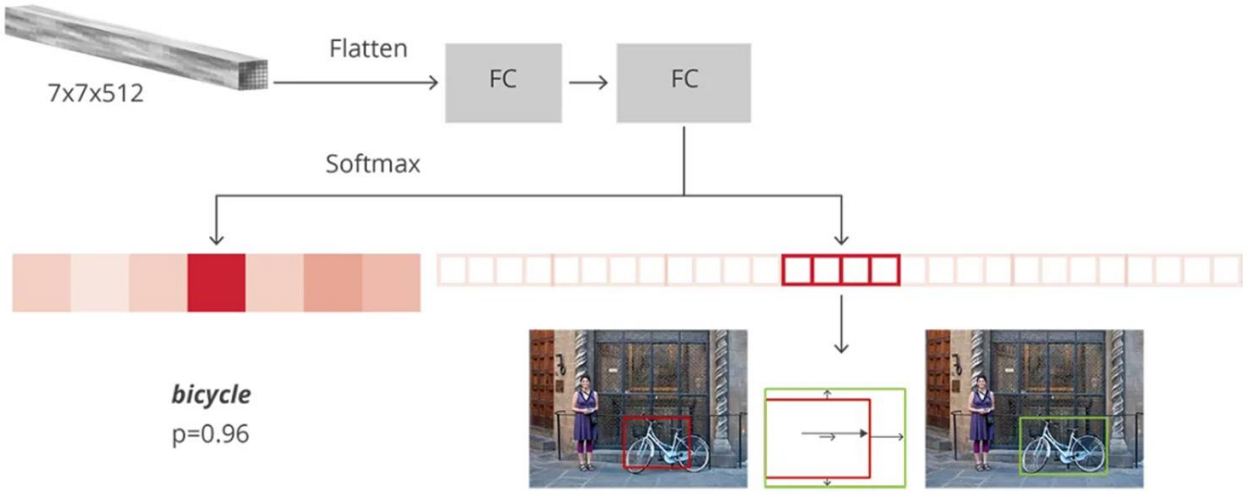


Figure 2.13 Region-based Convolutional Neural Network architecture

The double aims of classification and bounding box regression are critical to Faster R-CNN's efficacy since they ensure that each area suggestion is reliably identified and precisely localized.

The R-CNN module successfully converts the high-dimensional feature maps into meaningful classifications and bounding box adjustments by integrating these fully connected layers and the corresponding activation functions.

2.5 You Only Look Once (YOLO)

For real-time object detection, a significant improvement has been made with the introduction of the "You Only Look Once" (YOLO) algorithm proposed by Redmon et al. [19] What makes YOLO different is the usage of regression-based optimization to predict the bounding boxes and the class label in one training pass of the network. Unlike conventional object detection methods based on region proposals followed by classification, YOLO uses one-stage architecture to simultaneously predict several bounding boxes and their corresponding class probabilities [19]. This unified framework of YOLO allows it to achieve high computational efficiency and real-time performance.

Yolo algorithm is based on Darknet library, hence is an open-source project developed in C and CUDA.[20]. YOLO process - YOLO works by dividing the input image into a grid of cells, where each cell directly predicts a bounding box and confidence for that box [21]. for example, an input image of size (256 × 256) can be segmented into a (3 × 3) grid for instance, shown in Figure 2.14.

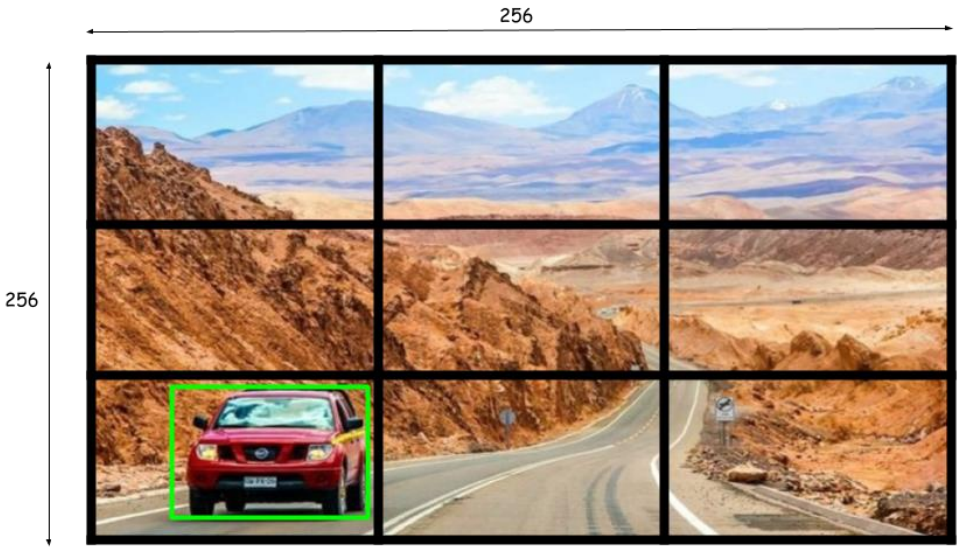


Figure 2.14 YOLO using Grid (3 x 3)

Additionally, YOLO also has an output tensor of size (9 × (c+5)), by dividing the image into 9 shown in Figure 2.15Figure 2.15, one corresponding to each of the bounding boxes in the grid cells

containing the center coordinates, dimensions and the objectness and class confidence scores for the bounding boxes within that cell [22].

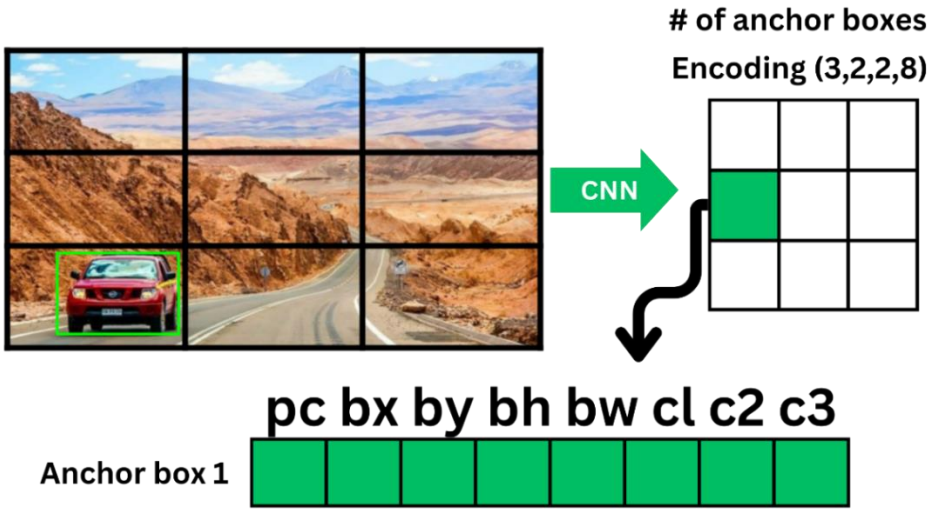


Figure 2.15 Output structure in YOLO

This Grid based image processing approach is the main reason that makes YOLO to perform detection in Real-time and that is why it is considered as a weightless solution that can be used in applications where you need to identify the objects quickly, namely autonomous vehicles, surveillance systems, and robotics.

The YOLO model outperforms earlier algorithms in terms of speed and optimization. Its exceptional performance of 45 frames per second is because of its novel architecture, which uses a single neural network to execute all the task's aspects. A visual performance comparison of YOLO and R-CNN is shown in Table 2.1, showing the latter's higher efficiency.

Table 2-1 Comparison between R-CNN and YOLO

Model	FPS (Frames per second)	Real-time
Fast YOLO	155	OUI
YOLO	45	OUI
YOLO VGG-16	21	NON
FAST R-CNN	0.5	NON
FAST R-CNN VGG-16	7	NON
FAST R-CNN ZF	18	NON

2.5.1 Architecture

YOLO's architecture is simple and based on basic layer types, for example convolution layers with 3x3 kernels and max-pooling layers with 2x2 kernels. However, it does not use completely connected layers, increasing its computational effectiveness even further. The model has nine convolutional layers with parametric ReLU activations, followed by a max-pooling layer that shrinks the input image size from 416x416 to 13x13. This 13x13 grid serves as the foundation for object detection, with each cell tasked with forecasting bounding boxes and class probabilities (Bounding box scores below thirty percent are eliminated) [23].

The YOLO model produces a tensor of size 13x13x125, with each grid cell represented by 125 channels. These channels encode data about bounding boxes and class predictions, including:

- **Box definition:** made up of (x, y, width, height, "is the object" confidence). The x and y coordinates represent the bounding box's centroid with the grid cell position.
- **Class probabilities:** considered only if there is a high level of confidence in "is the object.".

2.5.2 Evolution of YOLO

2.5.2.1 YOLOv1

YOLOv1 revolutionized object detection by framing it as a regression problem, departing from traditional classification methods. It employed a single CNN to detect objects, dividing the input image into a grid and making multiple predictions per cell. These predictions included bounding boxes, confidence scores, and class probabilities. YOLOv1 then filtered out low-confidence predictions and removed overlapping boxes to generate the final output. This innovative approach enabled real-time object detection with remarkable speed [24].

2.5.2.2 YOLOv2

Building upon YOLOv1's success, YOLOv2 introduced several key advancements. Batch normalization was incorporated into all convolutional layers, enhancing stability and performance while reducing overfitting. The model's ability to handle high-resolution images improved its accuracy in detecting smaller objects. YOLOv2 adopted anchor boxes, inspired by Faster R-CNN, which significantly improved the accuracy of predicting object shapes and sizes. These enhancements solidified YOLOv2's position as a leading real-time object detection algorithm [21(p. 900),24].

2.5.2.3 YOLOv3

YOLOv3 launched a new backbone network, Darknet-53, that made use of residual connections. It also made various design improvements to boost accuracy while remaining fast. At 320x320 resolution, YOLOv3 ran in 22 ms with 28.2 mAP. It achieved 57.9 mAP@50 in 51 ms on a Nvidia Titan X, compared to RetinaNet's 57.5 mAP@50 in 198 ms, which was comparable but 3.8x quicker. Figure 2.16 shows YOLOv3 upsampling (decoding) [24].

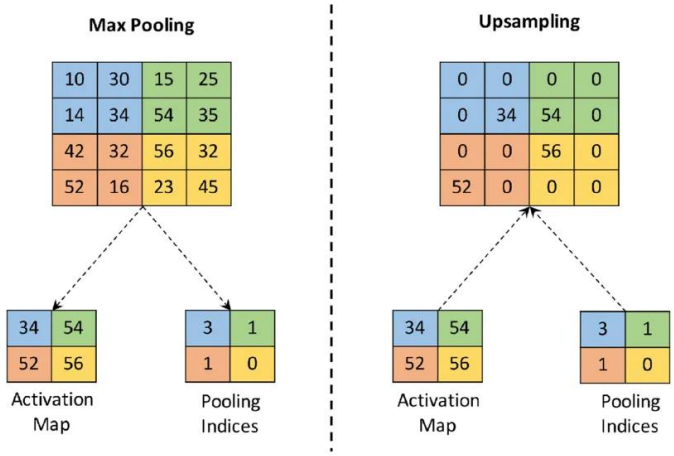


Figure 2.16 Upsampling YOLOv3

2.5.2.4 YOLOv4

YOLOv4 offered various new ways to increase accuracy and speed. It improved accuracy by using a CSPDarknet backbone (convolutional neural network and object detection) and introducing novel approaches like spatial attention, the Mish activation function, and GIoU (Generalized IoU) loss 3. The updated YOLOv4 method showed a 0.5% gain in average precision (AP) compared to the previous approach while lowering the model's weight file size by 45.3 M (number of total parameters) [24].

2.5.2.5 YOLOv5

YOLOv5 included more modifications to improve precision and efficiency. It used a Scaled-YOLOv4 backbone and novel methods, including CIoU (Complete IoU) loss function and CSPDarknet53-PANet-SPP. This model structure is composed of CSPDarknet-53, Spatial Pyramid Pooling in Deep Convolutional networks (SPPnet), Path Aggregation Network (PANet), to improve accuracy [24]. Yolov5 Structure looks bit different then the olde versions shown in Figure 2.17 .

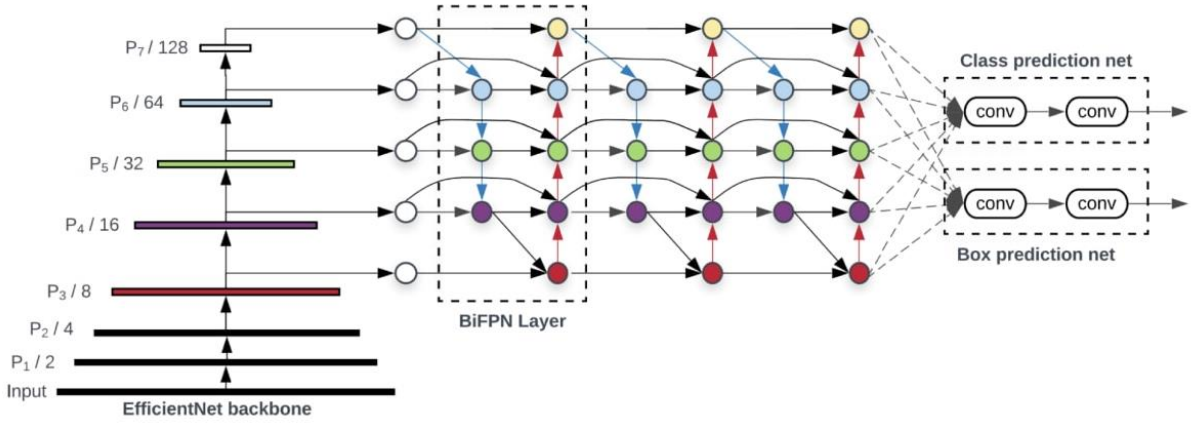


Figure 2.17 Structure of YOLOv5

The updated YOLOv5 algorithm improved mean average accuracy (mAP) by 0.7% over YOLOv4 while reducing model weight file size by 53.7 M (parameters). These enhancements improved YOLOv5's effectiveness and efficiency as a real-time object identification tool.

2.5.2.6 YOLOv6

YOLOv6 used a CSPDarknet-X backbone and incorporated novel ways to improve accuracy, including panoptic segmentation, the Swish activation function, and DIoU (Distance IoU) loss function. The YOLOv6 came with new Framework shown in Figure 2.18.

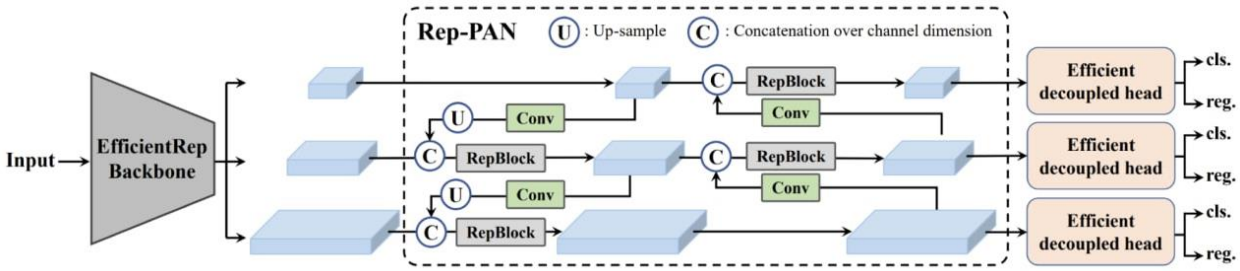


Figure 2.18 Framework of YOLOv6

The updated YOLOv6 algorithm improved average accuracy (AP) by 0.8% over YOLOv5 while reducing the model's weight file size by 60.2 M (parameters). These developments increased the power of YOLOv6's real-time object-detecting capabilities [24].

2.5.2.7 YOLOv7

In its design, YOLOv7 used a CSPDarknet-Z backbone. To improve accuracy, the YOLOv7 object detection algorithm was adjusted using unique approaches, for example object-centric segmentation, a Leaky ReLU (Rectified Linear Unit) activation function, and TIoU(Triplet) loss.

The updated YOLOv7 algorithm improved average accuracy (AP) by 1.0% over YOLOv6, while also lowering the model's weight file size by 70.5 M. These enhancements made the YOLOv7 object detection algorithm an even more effective tool for real-time object recognition[25].

2.5.2.8 YOLOv8

YOLOv8 has introduced a new backbone architecture, the CSPDarknet-AA, which is an upgraded version of the CSPDarknet series noted for its efficiency and performance in object detection tasks.

One important approach added to YOLOv8 is multi-scale object detection. This method enables the model to detect items of varying sizes within a picture. Another notable improvement in YOLOv8 is the use of the ELU activation function. ELU, or Exponential Linear Unit, accelerates learning in deep neural networks by addressing the vanishing gradient problem, resulting in quicker convergence. Figure 2.19 shows the timeline release for YOLO [24].

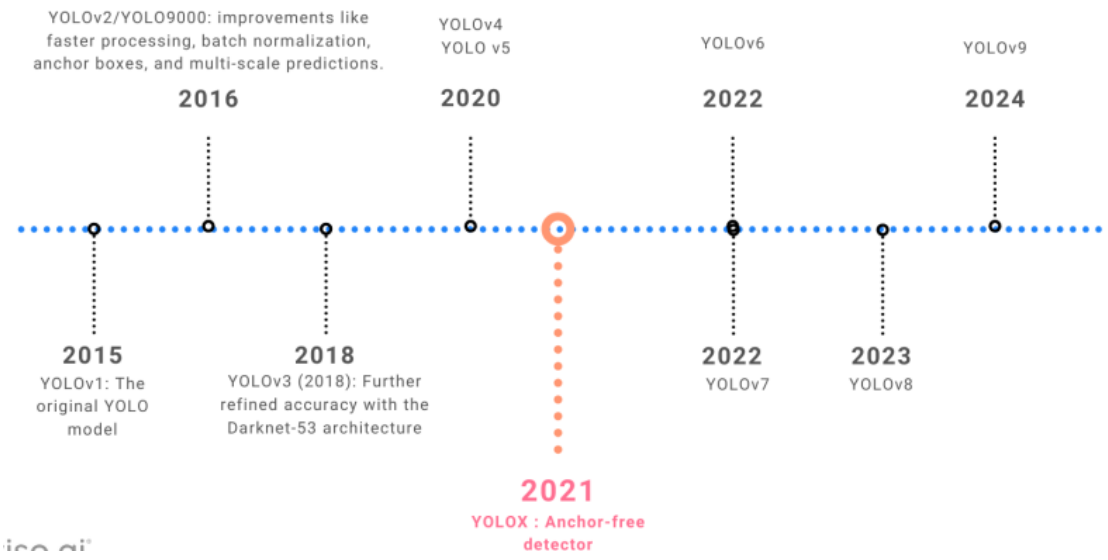


Figure 2.19 Yolo release timeline

YOLOv8 has incorporated the GIoU loss. GIoU, or Generalized Intersection over Union, is a more advanced variant of the IoU (Intersection over Union) metric that considers the form and size of the bounding boxes to improve object localization precision.

The YOLOv8 algorithm has a 1.2% increase in average precision (AP) over the YOLOv7, which is a substantial acquire. It accomplished this while decreasing the model's weight file size by 80.6 M (parameters), making it more efficient and deployable in resource-constrained scenarios.

2.6 Conclusion

This chapter has outlined the methodological framework for our ANPR system, from data collection to model evaluation. We detailed the data acquisition process, including preprocessing and labeling, and discussed the fundamentals of object detection with key performance metrics such as Intersection over Union (IoU) and Average Precision (AP). We examined the architectures of Faster R-CNN and YOLO, comparing their strengths and weaknesses. Ultimately, we chose YOLOv5 for its superior speed and suitability for deployment on single-board computers (SBC) like Raspberry Pi and Orange Pi, making it ideal for real-time applications. This comprehensive methodology establishes a solid foundation for developing effective ANPR systems.

Chapter 3: Comparative Analysis of Object Detection Models

3.1 Introduction

The performance of an object detection model in ANPR systems is primarily governed by two key metrics: accuracy and inference speed. Accuracy is crucial for ensuring reliable license plate detection, as any errors in this stage can propagate and degrade the overall ANPR performance. Inference speed, on the other hand, is a critical factor for real-time applications that require fast processing times, such as traffic monitoring, security systems, and intelligent transportation. By evaluating the trade-offs between these two factors, we aim to provide valuable insights to guide the selection of the most appropriate object detection model for ANPR deployments.

Besides the object detection component, the accuracy of Optical Character Recognition (OCR) models is equally vital for the success of ANPR systems. The OCR stage converts the detected license plate regions into machine-readable text, which is the ultimate output of the ANPR pipeline. Therefore, we also assess the performance of three OCR models –EasyOCR, YOLOv5, and Tesseract OCR– on a set of Algerian license plate examples, highlighting their respective strengths and limitations.

3.2 Models Analysis

This section presents the results got from applying Faster R-CNN and various versions of YOLO (including YOLOv5 and YOLOv8) for license plate detection. The analysis focuses on comparing the performance of these models across different metrics, including accuracy, precision, recall, inference, and detection speed in real time. Additionally, the impact of different hyperparameter settings and data augmentation techniques on the performance of each model is explored.

3.2.1 Faster R-CNN Trained Model

This section describes how to train a Faster R-CNN model for license plate detection using VGG16. Different epochs (turns) are used to assess the model's performance.

The Faster R-CNN model was trained using the VGG16 pre-trained network as the feature extractor. The model was trained on a dataset of cars images containing license plate sizes (640x640) shown in Figure 3.1.

Figure 3.1 labeled image

The VGG16 network was then initialized with pre-trained weights, while the region proposal network (RPN) and classifier heads were randomly initialized. This was followed by training using stochastic gradient descent (SGD) at a momentum of 0.9 and a learning rate of 0.001. To update the model parameters iteratively using estimated loss functions, an iterative technique of traversing the dataset was employed.

After every epoch, the model's performance was assessed on a different validation set, and measures alike average precision (AP), precision, recall, and inference speed were carefully examined.

3.2.1.1 Results and Discussion

The training routine spanned 59 epochs, wherein the model exhibited consistent performance improvement, as explained in Table 2.2. The average precision (AP) surged from 0.68 at epoch 19 to 0.72 at epoch 30, eventually culminating at 0.81 by epoch 59. Despite these advancements, the model fell short of achieving perfect accuracy in license plate detection.

Table 3-1 Faster R-CNN average precision with different epoch values

Epoch	Batch Size	AP
19	9	0.68
30	9	0.72
59	9	0.81

After training, we used a webcam to evaluate the model's real-time performance. The model displayed an inference speed of 5 to 15 frames per second (fps), representing its suitability for real-time picture processing. Unfortunately, this shows that Faster R-CNN is slow in real-time object detection, especially with our SoC.

In this section, it's important to note that our primary aim was educational. We employed the Faster R-CNN model to deepen our understanding of object detection methodologies. While the model's real-time performance may not meet practical standards, our focus was on learning and experimentation rather than using it for the ANPR system.

3.2.2 YOLOv8

The training procedure for our model that using YOLOv8 was thoroughly monitored, with the model's performance (specifically fps) evaluated at various phases of the training. The findings attained during this process were quite encouraging, demonstrating the excellent capabilities of the YOLOv8 architecture for license plate detection.

At the 20^{th} epoch, our model had already achieved an impressive average accuracy (AP) of 0.90 on the validation set. This means that the model could accurately identify 90% of the license plates in the test dataset. This early-stage result shows the model's remarkable inclination for learning the visual properties of license plates, even with a relatively small number of training cycles (epoch).

As the training advanced until the 30^{th} epoch, the model's average accuracy continued to increase dramatically. The AP increased to 0.99, confirming that the model could now accurately identify 99% of the license plates in the test dataset. This level of accuracy is extremely desirable for real-world Automatic Number Plate Recognition (ANPR) applications, where the ability to detect and localize license plates is of the utmost importance.

Finally, after 59 epochs of training, the YOLOv8 model attained a flawless AP of 0.99 on the test dataset. This shows that the model could successfully detect every single license plate in the test photos, displaying its excellent robustness and reliability. This impressive accuracy proves the model's capacity to handle a wide range of tough settings, including differences in lighting conditions, occlusions, and varied license plate styles.

The constant improvement in the model's performance 40 throughout the training phase, culminating in the ideal 0.99 AP at the 59th epoch, is a testament to the power and versatility of the YOLOv8 architecture. we use 300 epoch and the accuracy is the same 0.99, Figure 3.2 show how this model performs on training with 300 epoch.

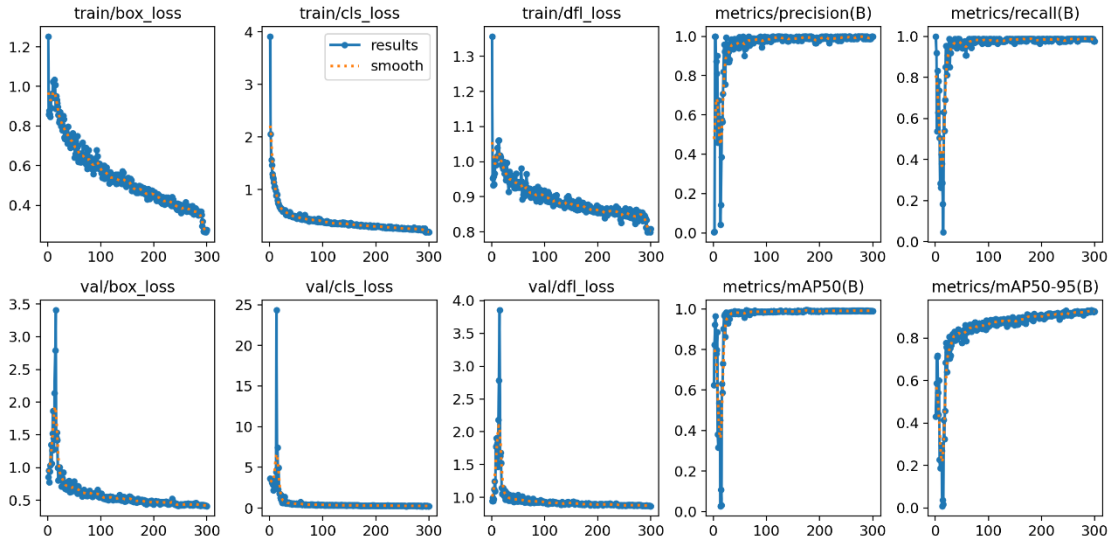


Figure 3.2 YOLOv8 Training Graphs (300 epoch)

We employed the YOLOv8 model to detect license plates in real-time and observed an impressive inference speed ranging between 30 and 45 frames per second (fps). This high frame rate underscores the model's efficiency and its suitability for real-time applications, making it highly effective for ANPR systems where rapid and accurate license plate detection is crucial. To see the performance of YOLOv8, we gave to the model new pictures, and the result shown in Figure 3.3.

Figure 3.3 licence plate detection using Yolov8

3.2.3 YOLOv5

The application of the YOLOv5 object detection model specifically trained for license plate detection on resource-constrained devices. The study analyzes the model's training process, performance, and suitability for real-world ANPR (Automatic Number Plate Recognition) on boards like Raspberry Pi and Orange Pi.

The training process for the model resulted in notable performance improvements, especially in speed (fps). The Average Precision (AP) on the validation set increased consistently throughout the training epochs, reaching values of 0.86 at epoch 18, then 0.98 at epoch 29, and culminating in a near-perfect score of 0.99 at epoch 59. These results highlight the model's effectiveness in precisely detecting and localizing license plates through a variety of scenarios. All the graphs are shown in Figure 3.4.

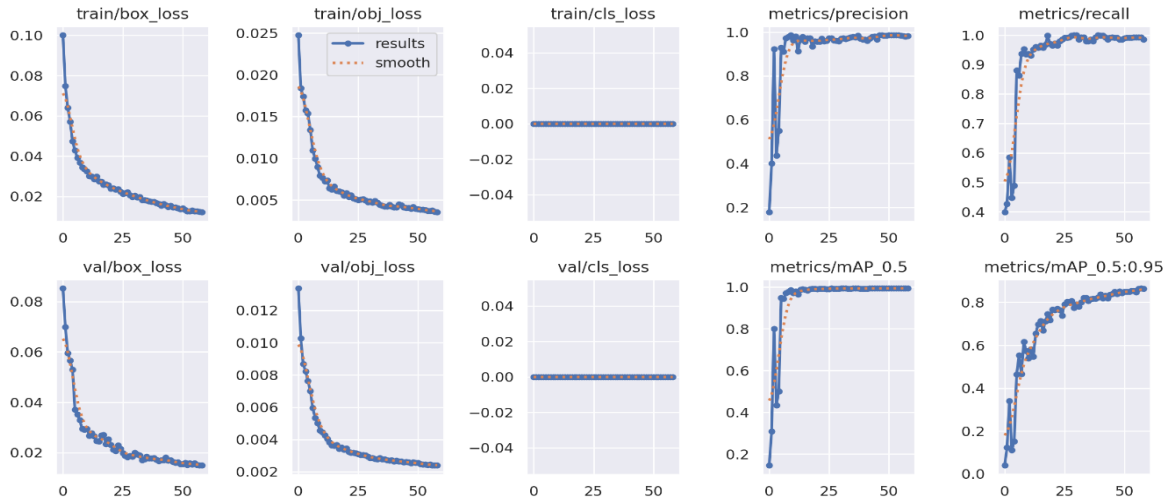


Figure 3.4 Train results/graphs with 59 epoch (YOLOv5)

Furthermore, the model's inference speed was rigorously evaluated. The average processing time per image was assessed on both a Raspberry Pi 4 and an Orange Pi 4, with the model achieving an impressive speed of 25 frames per second (fps) on the Raspberry Pi 4B and 30 fps on the Orange Pi 5B using a neural processing unit (NPU). This level of performance shows that the model is well-suited for real-time Automatic Number Plate Recognition (ANPR) applications on devices constrained by limited computational resources. A sample of license plate prediction is shown in Figure 3.5.

Figure 3.5 License plate prediction

The selection of YOLOv5 for this project was a deliberate and strategic choice. YOLOv5 is renowned for its optimal balance between accuracy and speed, rendering it ideally suited for deployment on devices with restricted computational capabilities. Moreover, its open-source nature and extensive community support enhance its application for developers focused on embedded systems.

3.2.4 Models comparison

The comparison reveals the higher performance of YOLOv8 in terms of both accuracy and inference speed, making it an excellent candidate for real-time object detection applications. YOLOv5 also offers a solid balance between accuracy and speed, making it ideal for deployment on machines with low processing resources. In contrast, Faster R-CNN, while accurate, may not be suited for real-time applications because of its slower inference speed. These insights are critical for making judgments when selecting models for embedded systems. This comparison is summarized in the table below.

Table 3-2 The performance of Faster R-CNN, YOLOv5, and YOLOv8 across different training epochs

Model	Batch Size	Epoch	AP	FPS (max)
Faster R-CNN	09	19	0.68	**5-10**
		30	0.72	
		59	0.81	
YOLOv5	09	18	0.86	**30**
		29	0.98	
		59	0.99	
YOLOv8	09	20	0.90	**45**
		30	0.99	
		59	0.99	

3.3 Number Plate Recognition

Optical Character Recognition (OCR) technology is crucial for number plate detection. This process involves capturing and preprocessing images of license plates to enhance quality, followed by using OCR algorithms to extract the numeric characters. OCR systems must be customized to recognize the specific fonts and styles of Algerian number plates (cars with digits from 0 to 9). The primary challenge is achieving high accuracy under varying lighting conditions, angles, and obstructions. For advanced OCR models, we tried YOLOv5 and other pre-trained libraries, like EasyOCR and Tesseract OCR.

3.3.1 Training OCR model

To effectively recognize Algerian license plate digits, we trained the YOLOv5 model on a dataset specifically curated for this aim. This dataset is composed of multiple photos with individual digits collected from Algerian license plates, tagged with appropriate digit values (0 to 9).

Algerian license plate numerals face unique issues because of variances in font styles, widths, and spacing. To solve these problems, we used particular tactics during the training process:

1. Data Augmentation: The dataset was enhanced using techniques including random cropping, flipping, and brightness modifications to boost its diversity and improve the model's generalization capacity. This meant that the model could recognize digits reliably, even when faced with differences in appearance.

2. Transfer Learning: We employed pre-learned weights from YOLOv5 models trained on large-scale datasets. This transfer learning strategy exploited the knowledge learned from prior training, expediting the training process and boosting the model's performance on the specific task of recognizing Algerian license plate numbers.

3. Hyperparameter Optimization: We carefully tweaked the hyperparameters of the YOLOv5 model, such as learning rate and batch size, to improve its performance for the given dataset and job. This fine-tuning approach guaranteed that the model reached the best possible accuracy and efficiency.

3.3.1.1 Performance analysis and evaluation

The trained YOLOv5 model was assessed on a second test dataset containing unseen photos of Algerian license plate digits. The model's performance was measured using criteria like accuracy,

precision, recall, and inference speed. Results revealed that YOLOv5 exhibited great accuracy (0.95 AP) in recognizing the digits shown in Figure 3.6, even in demanding situations with varied illumination, angles, and font styles. Additionally, YOLOv5 provides a strong FPS in real-time, making it ideal for real-world applications.

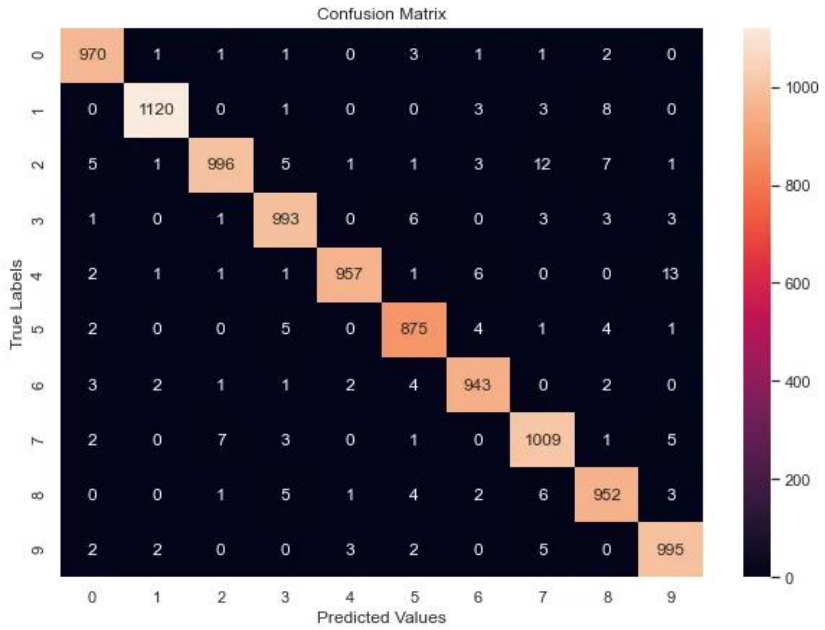


Figure 3.6 YOLOv5 OCR confusion matrix

YOLOv5 has proven to be an effective tool for optical character recognition on Algerian license plate digits. Through targeted training on a specialized dataset, data augmentation techniques, and hyperparameter optimization, the model achieved excellent accuracy in detecting the digits 0 to 9, even in adverse settings. This illustrates the potential of YOLOv5 for robust and efficient Automatic Number Plate Recognition (ANPR) systems in Algeria, contributing to improved traffic management, law enforcement, and security applications.

3.3.2 Comparison

Optical Character Recognition (OCR) technology has emerged as a cornerstone in automatic number plate recognition (ANPR), particularly for applications requiring the detection and interpretation of automobile license plates. When setting Algerian license plates, which typically display digits from 0 to 9, choosing an OCR model is critical. This study focuses on evaluating three famous OCR models—YOLOv5, EasyOCR, and Tesseract OCR—in terms of their efficacy, accuracy, and applicability for real-time applications.

YOLOv5, a state-of-the-art object identification model, is famous for its high accuracy and real-time performance. Its ability to be trained on individual datasets makes it a strong tool for specialized OCR jobs, especially those involving the distinctive peculiarities of Algerian license plates. However, this precision comes with a cost: large computational resources and well-curated training data are necessary to attain optimal outcomes.

EasyOCR: stands out for its user-friendly interface and comprehensive linguistic support. Its ease of use makes it an attractive option for developers who require a rapid and efficient OCR solution. Nevertheless, EasyOCR may find restrictions in accuracy and customization, particularly when working with extremely unique or localized datasets such as Algerian license plates.

Tesseract OCR: an open-source project supported by Google, offers a high accuracy and versatility because of its flexibility and support for sufficient languages. It is extremely adaptive for various OCR tasks. However, Tesseract's requirement for significant configuration and its relative slowness in real-time applications may pose issues for certain use cases, notably those seeking instantaneous results.

The intention of this comparison study is to analyze the performance of these three OCR models in Algerian license plate recognition shown table. By assessing the trade-offs in terms of accuracy, convenience of use, customization, and real-time application, this research intends to provide a thorough assessment that will aid practitioners in selecting the most effective OCR solution for their individual needs. Through thorough testing and analysis, we attempt to highlight the strengths and limitations of each model, ultimately contributing to the improvement of OCR technology in specialized applications.

Table 3-3 Comparison between different OCR models

number	YOLOv5	EasyOcr	Tesseract OCR
	0549937309	0549932309	0549977309
	00462932216	00462932216	00462932216
	209871126	209871126	2098711176

As can be seen from the table, EasyOCR correctly recognized all three license plate numbers. YOLOv5 also correctly recognized two out of the three license plate numbers, with one error in the first number. Tesseract OCR correctly recognized the first two license plate numbers, but made an error in the last number.

This comparison suggests that EasyOCR is the most accurate OCR model for Algerian license plate recognition. YOLOv5 is also a good option, but it may be less accurate than EasyOCR. Tesseract OCR is the least accurate of the three models, but it is still a viable option for some applications.

While EasyOCR can be slower than other OCR models without a GPU, using a GPU with CUDA support can significantly improve its performance. If you have access to a GPU, it is highly recommended to use it for optimal results. However, even without a GPU, there are still ways to improve EasyOCR's performance by choosing a smaller model, reducing the image resolution, and optimising your code.

3.4 Conclusion

The analysis reveals that YOLOv8 outperforms the other object detection models in both accuracy and inference speed, making it the most suitable choice for real-time ANPR. YOLOv5 also shows a solid balance between the two factors, making it a practical alternative for resource-constrained deployments. In contrast, while Faster R-CNN is accurate, its slower inference speed may limit its applicability in real-time ANPR systems.

The evaluation of OCR models shows that Easy OCR is the most accurate for Algerian license plate character recognition, followed by YOLOv5 and Tesseract OCR. These findings provide valuable guidance for researchers and developers in selecting the optimal combination of object detection and character recognition models to achieve high-performance, real-time ANPR solutions.

Chapter 4: Implementation

4.1 Introduction

This chapter delves into the practical issues of building an Automatic Number Plate Recognition (ANPR) system. The technical specifics, including the hardware and software components necessary for the system's operation, are outlined at the beginning of the chapter. A special focus is placed on the Orange Pi, an important piece of hardware, explaining how it is established and integrated into the system. The implementation of the system is then examined, with an emphasis on the creation of the user interface and the operational process. This chapter offers a thorough overview of the implementation procedure, emphasizing the difficulties faced and the methods developed to produce a working ANPR system.

4.2 Technical details

This section covers the key technical aspects of our ANPR (Automatic Number Plate Recognition) system, built using programmable boards. We provide an overview of the software architecture and its components, detail the necessary hardware specifications, and describe the development environment, including the tools and frameworks used to create a robust and efficient system.

4.2.1 Software

4.2.1.1 Python

It is a powerful and high-level programming language that is becoming popular for academic and research work. Python is well-known for its readability, ease of use, and large library ecosystem. It has several benefits for both researchers and developers [26]. Its broad ecosystem of tools and frameworks is enhanced by its active community and robust standard library, making it more useful for a wider range of applications.

Python is extensively used in this project to create and deploy machine learning and deep learning models for artificial intelligence. Effective model training depends on the language's versatility and the availability of strong libraries like YOLO and TensorFlow. These libraries include

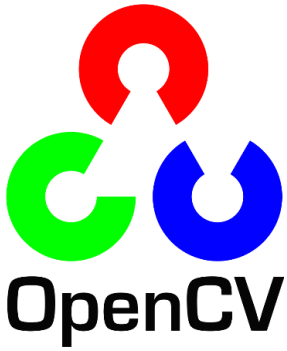
pre-built functions and algorithms that reduce the time and effort needed to construct a complicated AI model.

Furthermore, Python's capacity to manage enormous datasets and perform complex computations is improved by its robust support for scientific computing and data analysis, provided by libraries like NumPy and Pandas. This extensive toolkit guarantees that the AI models created are resilient, scalable, and efficient [27].

To accomplish the project's research objectives, I will go over the precise implementation details, how to use Python's features and libraries, and how to integrate the produced components in the sections that follow.

4.2.1.2 OpenCV

OpenCV (Open-Source Computer Vision Library) is a computer vision and machine learning framework. It is an advanced and popular open-source library that offers a comprehensive collection of tools for real-time image and video processing [28]. It was the perfect option for creating the camera on screen display and the user interface for the present project because of its cross-platform compatibility and interaction with the GTK (GIMP Toolkit) library [29].



We could successfully record video from the camera with OpenCV and play it back in a graphical user interface built with GTK. Thanks to the rich capabilities of the OpenCV package, you can use a variety of computer vision algorithms, like object identification, facial recognition, and image classification, on the recorded video. By integrating OpenCV and GTK, we could design a user-friendly interface that allowed for smooth interaction with our project's camera-based features.

During the implementation process, I made use of OpenCV's extensive pre-built algorithms and functions, lively community, and comprehensive documentation. This allowed us to fast prototype and improve the project's essential features while emphasizing the special qualities and original contributions of this study.

4.2.1.3 Firebase

It is a complete platform created by Google for developing and managing websites and mobile apps. Firebase, which is well-known for its real-time database capabilities, is a great option for projects needing strong backend support because it provides a range of tools and services that expedite the development process [30].

A comprehensive database for drivers and their IDs is created using Firebase. Because of its real-time database, which enables effective data storage and retrieval, the application is always responsive and up-to-date. Furthermore, integrating Firebase with additional services like cloud storage and authentication improves the system's scalability and security [31]. The implementation of Firebase in this project will be covered in the sections that follow. These will include information on the database's structure, data management techniques, and overall impact on the project's goals.

4.2.2 Hardware

4.2.2.1 Single-Board Computers (SBCs)

Compact, fully working computers constructed on a single circuit board are known as single-board computers or SBCs. They incorporate every crucial part of a conventional computer, such as memory, storage, input/output (I/O) connections, and a CPU. SBCs are well known for being inexpensive, flexible, and simple to use, which makes them perfect for embedded systems, prototyping [32], and educational applications. This chapter examines the features, capabilities, and applications of three well-known SBCs: the Raspberry Pi, Orange Pi, and Jetson Nano.

4.2.2.2 Raspberry Pi

One of the most well-known SBCs is the Raspberry Pi, which was created by the Raspberry Pi Foundation. Although its original goal was to advance computer science education, its low cost and strong community support have led to its adoption for a variety of uses [33]. as is shown in Figure 4.1 the least Raspberry pi board.

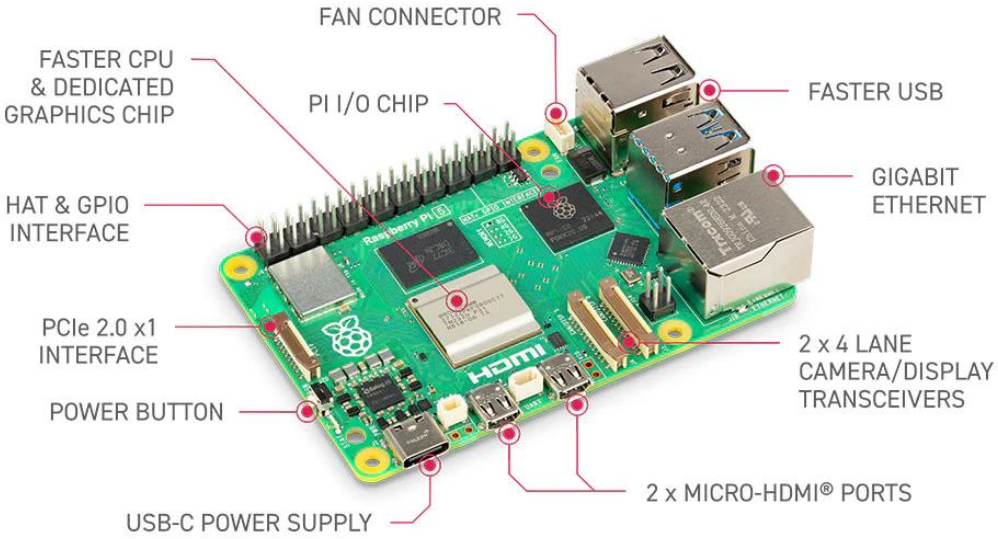


Figure 4.1 Raspberry Pi 5 specifications

Uses:

- Teaching resources for electronics and programming
- Automation systems for homes
- Media centers Internet of things

Advantages and Drawbacks: The advantages of the Raspberry Pi are its low cost, wide user base, and thorough documentation. But when compared to more potent SBCs, its performance is constrained, which makes it less appropriate for demanding applications.

4.2.2.3 Overview of Orange Pi

Orange Pi, created by Shenzhen Xunlong Software, is a line of SBCs with a similar design to the Raspberry Pi, but with more capabilities and sophisticated features. Orange Pi wants to give developers and enthusiasts affordable options [34].

Uses:

- Applications for multimedia
- Automation of the home
- Creation and testing of embedded and Internet of Things systems

Advantages and Drawbacks: At a comparable cost to Raspberry Pi, Orange Pi boards have comparable specs and more capabilities. The user base is limited, though, and there may not be as much support or documentation [35].

4.2.2.4 Jetson Nano

The Jetson Nano is an SBC made by Nvidia that is intended primarily for AI and machine learning applications. Its small physical footprint and potent GPU capabilities make it ideal for edge computing and AI-driven applications [36]. Figure 4.2 shows jetson nano board.

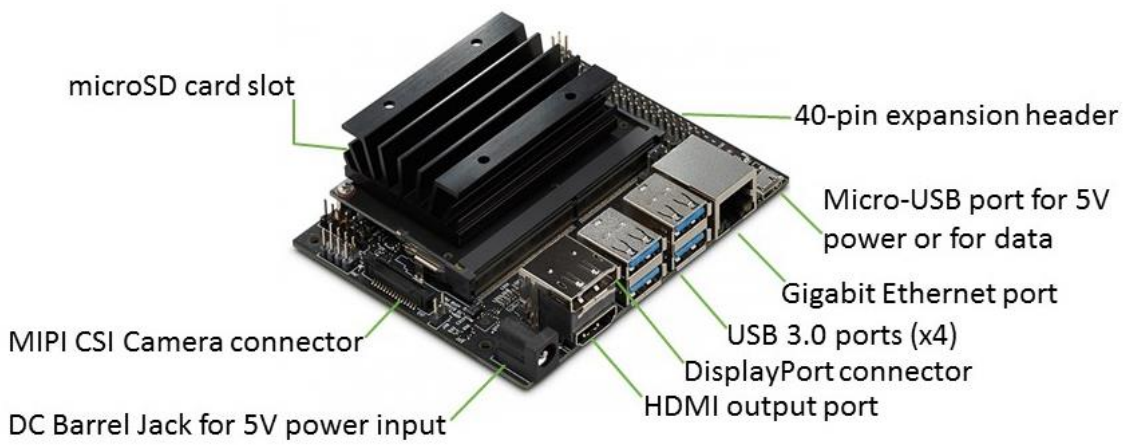


Figure 4.2 Jetson Nano Developer Kit (I/O)

Uses:

- Deep learning and AI initiatives
- Computer vision
- Cutting-edge robotics

Advantages and drawbacks: Jetson Nano's strong GPU makes it an excellent choice for AI and machine learning jobs. Its primary drawbacks are the additional complexity involved in developing GPU-accelerated applications and its greater price when compared to other SBCs.

4.2.3 Orange Pi

The Orange Pi 5B uses the Rockchip RK3588S, a new generation octa-core 64-bit ARM processor that comprises four quad-core A76 and four quad-core A55. It uses Samsung's 8nm LP process technology, has a large-core main frequency of up to 2.4GHz, an integrated ARM Mali-G610 MP4 GPU, and is embedded with high-performance 3D and 2D image acceleration modules. It also has an AI accelerator NPU that can process up to six tops. The memory capacity of the processor is 4GB/8GB/16GB (LPDDR4/4x) and 32GB/64GB/128GB/256GB onboard eMMC. The processor can process images up to 8K [37].

Many new interfaces are introduced by the Orange Pi 5B, including Type-C, HDMI output, Bluetooth, WiFi6, USB2.0, USB3.0 interface, Gigabit Ethernet connector, and a 26-pin expansion pin header. It can be extensively applied to a variety of AIoT industries, including high-end tablets, edge computing, cloud computing, AR/VR, smart security, and smart homes [37]. Figure 4.3 shows all the orange pi 5B interfaces.

Orange Pi OS, the official operating system created by Orange Pi, is supported by Orange Pi 5B. It also supports other operating systems, including Ubuntu 20.04, Ubuntu 22.04, Debian 11, and Android 12.1 [37].

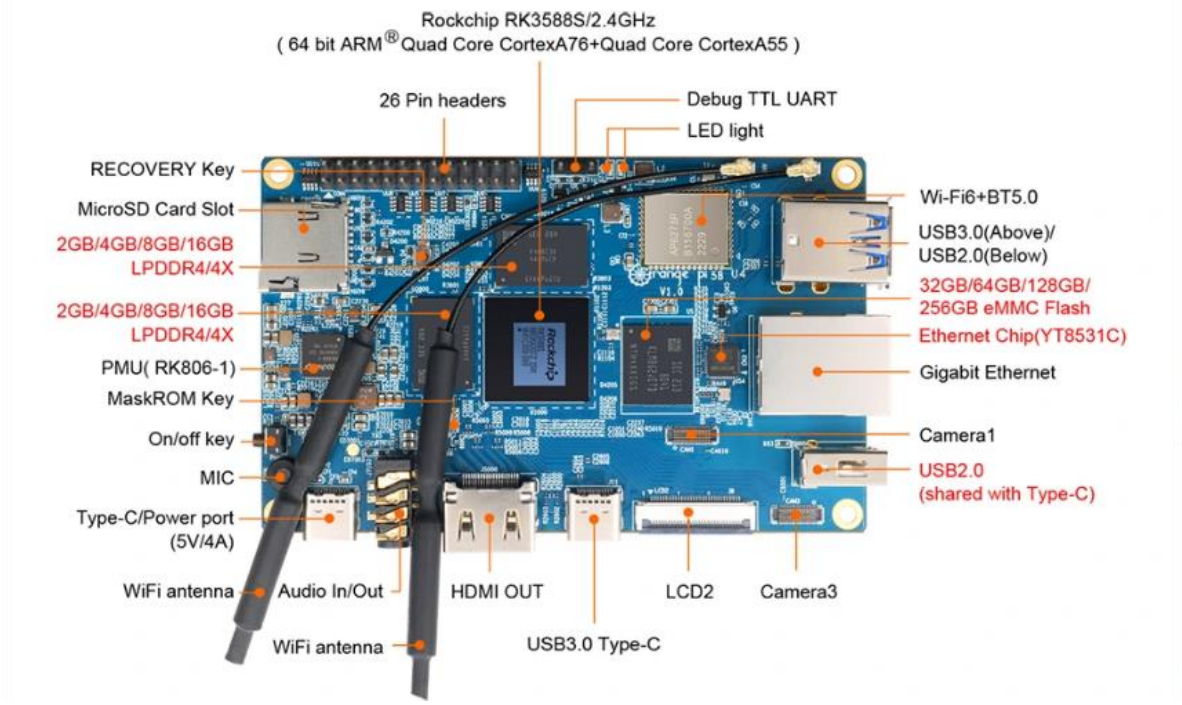


Figure 4.3 Orange Pi 5B (top view)

For more detail's Table 4-1 shows the Specification of Orange PI 5B.

Table 4-1 Hardware features of Orange Pi 5B

Introduction to hardware features	
CPU	• Rockchip RK3588S (8nm LP process) • 8-core 64-bit processor • 4-core Cortex-A76 and 4-core Cortex-A55 core architecture • The main frequency of the large core is up to 2.4GHz, and the main frequency of the small core is up to 1.8GHz.
GPU	• Integrated ARM Mali-G610 • OpenGL ES1.1/2.0/3.2, OpenCL 2.2 and Vulkan 1.2
NPU	• Built-in Al accelerator NPU with a computing power of up to 6 Tops • Support INT4/INT8/INT16 mixed operation
video output	• HDMI 2.1, up to 8K @60Hz • DPI.4 (DisplayPort) • 2 * MIPI ID-PHY TX 4Lane
Memory	4GB/8GB/16GB (LPDDR4/4x)
Camera	• 1 * MIPI CSI 4Lane • 2 * MIPI D-PHY RX 4Lane
PMU	RK806-1
onboard storage	• MicroSD (TF) Card Slot • 32/64/128/256 GB eMMC
Ethernet	10/100/1000 Mbps ethernet (YT853)
WIFI+BT	Onboard WI-F16+BT 5.0 module (AP6275P), supports BLE
Audio	• 3.5mm headphone jack audio in/out • Onboard MIC input • HDMI output
USB interface	1 * USB3.0 interface 2 * USB2.0 interface (one of which is shared with the Type-C interface) 1 * USB3.0 Type-C port
26pin extension header	Used to expand UART, PWM, 12C, SPI, CAN, and GPIO
Interfaces	
Debug serial port	3pin debugging serial port
LED	Power light and status light
Button	1 * MaskROM key, 1 * RECOVERY, 1 * switch key
Power supply	Type-C interface power supply 5V/4A;
Introduction of Appearance Specifications	
Product Size	100mm*62mm
Weight	46g

4.2.4 Prepare Orange PI

1- SD card, a class 10 (V10) or above high-speed with a minimum capacity of 8GB (32GB or above is recommended). In our case, we used SanDisk 128 GB class 10.

2- SD card reader.

3- HDMI to HDMI cable is used to connect the development board to an HDMI monitor or TV for display.

4- Power adapter 20W (Figure 4.4).

Figure 4.4 Orange Pi 5B power supply

As Figure 4.5 shown where the usb-c power port is. Because there is a difference between the two ports, one for power and the other for video display.

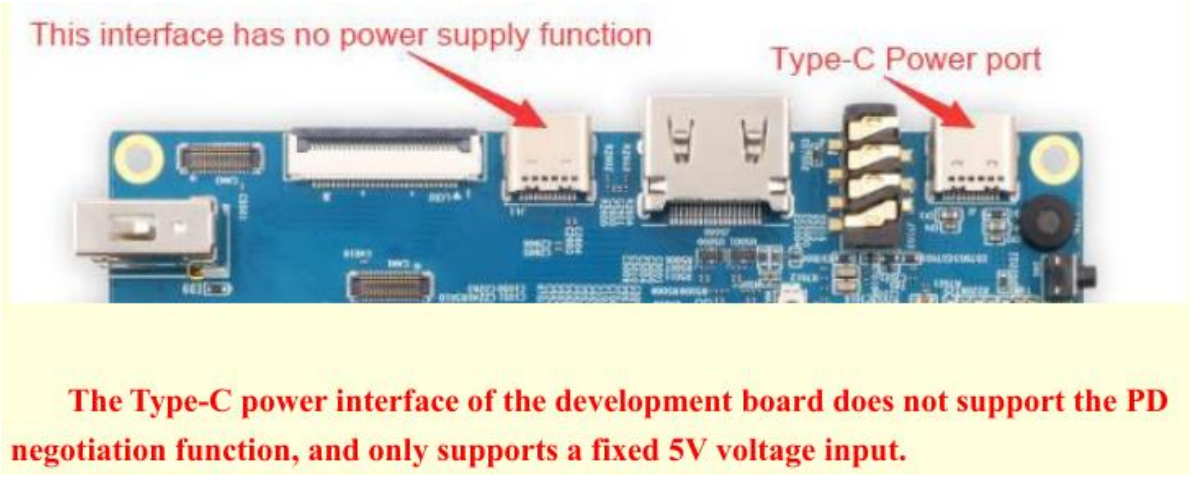


Figure 4.5 Orange Pi 5B bored for power port

4.3 Implementation

In this section, we delve deeper into the implementation of the ANPR system, focusing on the integration of electronic components and user interfaces. We will explore the hardware aspects, including sensors, cameras, and processing units.

4.3.1 User interface (Software)

The ANPR process begins with license plate detection, where the input image is first converted from RGB to gray-scale format to enhance the accuracy of subsequent recognition steps. Edge detection, segmentation, and feature analysis techniques are then employed to locate and isolate the license plate region within the image. And we can see how it works in Figure 4.6.

Figure 4.6 License plate detection

In designing the user interface for our ANPR system, our goal was to create a user-friendly and intuitive experience. The interface starts with plate detection and number recognition, ensuring accurate identification and interpretation of license plates. We developed four distinct interface states to enhance user interaction (Figure 4.7):

1- **Active State:** Implies the system is ready for detection.

2- **Detection State:** Provides detailed vehicle information, including type, driver's name and job, and the number of times the vehicle has entered the parking area.

3- **Marked State:** Alerts when the vehicle is marked for special conditions, such as restricted access, and ensures entry is denied if the vehicle is already marked.

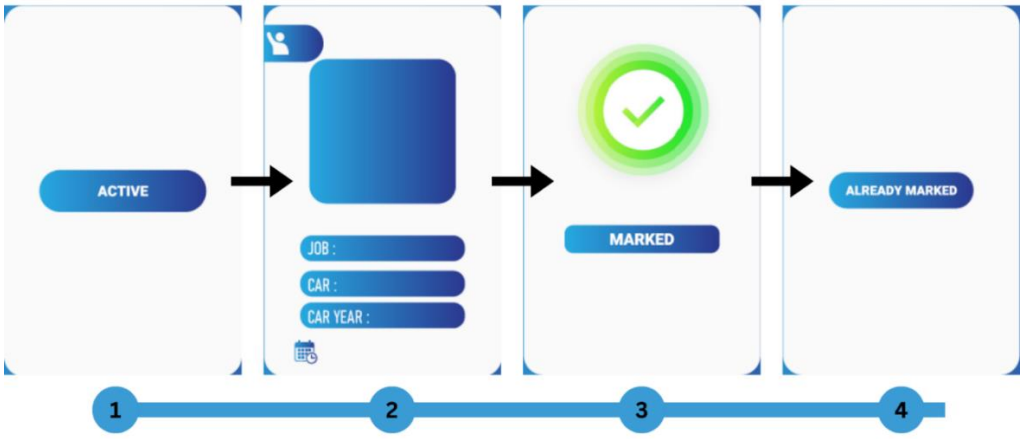


Figure 4.7 Interface Modes

As Figure 4.7 shows, the first state specifies when the system is ready for detection. Upon successful detection, the second state provides detailed information about the vehicle, including the type of vehicle, the driver's name and job, and the number of times the vehicle has entered the parking area. The third state alerts the user when the vehicle is marked for special conditions, such as restricted access. The fourth state ensures that entry is denied if the system has already marked the vehicle. Through these carefully designed interfaces, we strive to offer a seamless and informative user experience.

4.3.2 How it Works?

Firstly, we need to set up the hardware components, including a camera and an Orange Pi, as shown in Figure 4.8. This hardware setup is essential before implementing the software. Once the hardware is in place, the ANPR system can be easily started by launching the program from the software icon.

Figure 4.8 ANPR System (hardware components)

The application allows parking attendants or other users to manage vehicle entries efficiently by providing real-time information about the driver's name and car type at the time of entering. This functionality enhances operational efficiency and ensures a smooth entry process for approved vehicles. in Figure 4.9 we can see how application start first, showing the camera video and write that is active.

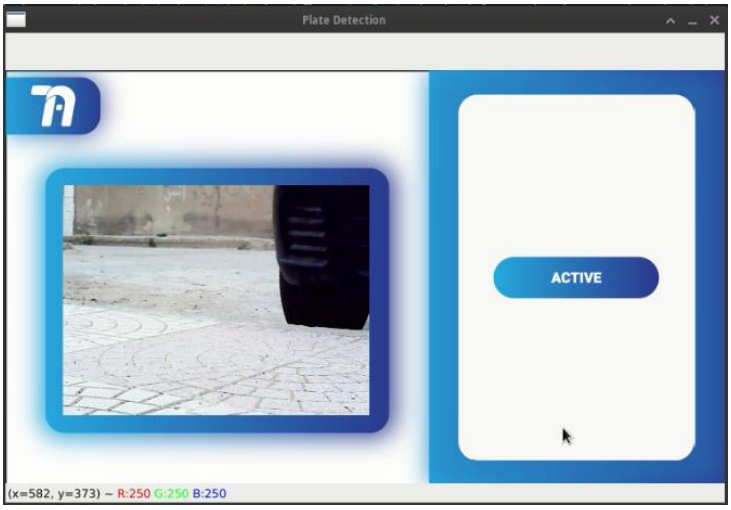


Figure 4.9 application first view

After recognizing the license plate, the application displays the vehicle's owner's name, car type, and time of attendance shown in Figure 4.9. This feature is essential for improving efficiency and security in a variety of contexts, including parking lots, gated communities, and corporate buildings. The application provides precise record-keeping and reduces human mistakes by automating the recognition process.

Figure 4.10 License Plate Detection App

The program will refuse admission and notify the user that the vehicle has already been marked if a car tries to enter again while the overall attendance limit is reached shown in Figure 4.11. This feature makes sure that capacity limits are followed, which keeps the area safe and orderly. A superuser can also alter the recorded attendance periods, giving flexibility for special circumstances or administrative changes. This feature facilitates the enforcement of attendance regulations and offers fine-grained control over access permissions.

Figure 4.11 Access Denied because of Full Attendance Limit and Recorded Entry

4.4 Conclusion

In conclusion, this chapter has discussed the mechanism of creating the ANPR system, focusing on the technical specifications and the integration of both hardware and software components. We have demonstrated the system's reality, including the user interface development and the underlying mechanics that enable license plate detection and recognition. This excellent implementation is a vital step towards developing a dependable and efficient system suitable for diverse applications, such as traffic control and law enforcement. The insights attained from this chapter pave the development of more advanced ANPR systems capable of working in different and demanding contexts. Additionally, while the user interface designed using GTK has been useful, it has been found to slow down the application. Future improvements may explore different graphical user interfaces to enhance performance and overall system efficiency.

Conclusion

Conclusion

Although many implementations of ANPR systems exist, the goal of this project was to build one that is specific for Algerian license plates. To achieve the defined objectives, this study addressed the development of an Automatic Number Plate Recognition (ANPR) system, focusing particularly on real-time applications.

First discussion established the stage by explaining the theoretical framework that underpins ANPR technologies, with a specific emphasis on deep learning algorithms tailored for real-time license plate recognition. The constraints inherent in this domain were investigated, as well as the applications that such a system could uncover across various Algerian industries.

The next chapters discussed the data collecting and preparation procedures that are critical to training a robust model. The diversity of data, particularly license plate variances, was highlighted as a crucial aspect in improving model accuracy. Cleaning and reformatting processes were also emphasized as critical parts in maintaining data integrity.

The essence of this project was the practical application of the ANPR system. An Orange Pi 5B served as the hardware foundation to support real time object detection. We described the architecture of the YOLO model and detailed its composition and its role in achieving high detection accuracy. in the deployment of the YOLOv5 model. This model demonstrated exceptional efficacy, achieving a detection accuracy of 99% and more than (25 FPS) in real-time conditions.

The performance of the system was evaluated using harsh metrics, such as confusion matrices and ROC curves. These analyzes confirmed the effectiveness of our approach and highlighted the resilience of our system to various operational scenarios. Then, we explored the applicability of the system in various areas, such as parking administration, vehicle tracking, access point management. The findings from this work contribute significantly to the field of ANPR, demonstrating the feasibility of such system to operate even under harsh conditions.

Despite the success achieved, we have identified areas that require improvement, especially regarding the user interface developed in GTK. Future work may address alternative interfaces to optimize performance and improve the performance of the system.

In conclusion, this study not only achieved its fundamental objectives but also provided new insights into the application of deep learning to ANPR systems. It demonstrated high accuracy and reliability in various categories, supporting the possibility of practical application. Future research approaches could include expanding the dataset to include more diverse license plate images and exploring advanced neural network models to further improve system performance. This achievement lays a solid foundation for subsequent advances in intelligent transportation management systems and paves the way for more widespread application of ANPR technology.

References

[1]: Rhead, M., Gurney, R., Ramalingam, S., & Cohen, N. (2012). Accuracy of automatic number plate recognition (ANPR) and real world UK number plate problems. *2012 IEEE International Carnahan Conference on Security Technology (ICCST)*, 286–291. https://doi.org/10.1109/CCST.2012.6393574

[2]: Gurney, R., Rhead, M., Lyons, V., & Ramalingam, S. (2013). The effect of ANPR camera settings on system performance. *5th International Conference on Imaging for Crime Detection and Prevention (ICDP 2013)*, 1–6. https://doi.org/10.1049/ic.2013.0276

[3]: Agostino, S. A. D., Shuldiner, P. W., Merrick, M., & Woodson, J. (1997). Real-time license plate reading for origin/destination studies. *Transportation Sensors and Controls: Collision Avoidance, Traffic Management, and ITS*, *2902*, 2–7. https://doi.org/10.1117/12.267134

[4]: Parsons, C. A., Savirimuthu, J., Wipond, R., & McArthur, K. (2012). ANPR: Code and Rhetorics of Compliance. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2141127

[5]: Kadhim, K. A., Mundher Adnan, M., Riyadh Waheed, S., & Alkhayyat, A. (2021). Automated high-security license plate recognition system. *Materials Today: Proceedings*, S2214785321034581. https://doi.org/10.1016/j.matpr.2021.04.533

[6]: Pawłowicz, B., Trybus, B., Salach, M., & Jankowski-Mihułowicz, P. (2020). Dynamic RFID Identification in Urban Traffic Management Systems. *Sensors*, *20*(15), Article 15. https://doi.org/10.3390/s20154225

[7]: AG, S. Fakhar, et al. (2019). Development of portable automatic number plate recognition (ANPR) system on Raspberry Pi. *International Journal of Electrical and Computer Engineering (IJECE)*, *9*(3), 1805. https://doi.org/10.11591/ijece.v9i3.pp1805-1813

[8]: *ANPR Parking Systems (Zatpark)*. (n.d.). Retrieved May 17, 2024, from https://zatpark.com/anpr-parking-systems/

[9]: Klingler, N. (2023, December 6). *Automatic Number Plate Recognition (ANPR) - 2024 Guide*. Viso.Ai. https://viso.ai/computer-vision/automatic-number-plate-recognition-anpr/

[10]: ANPR/ALPR systems in use Application areas of CARRIDA. (n.d.). *Carrida Technologies GmbH*. Retrieved May 17, 2024, from https://www.carrida-technologies.com/en/anpr-systems-in-use/

[11]: Berchmans, D., & Kumar, S. S. (2014). Optical character recognition: An overview and an insight. *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 1361–1365. https://doi.org/10.1109/ICCICCT.2014.6993174

[12]: Khan, M. A., Menouar, H., & Hamila, R. (2023). Visual crowd analysis: Open research problems. *AI Magazine*, *44*(3), 296–311. https://doi.org/10.1002/aaai.12117

[13]: Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2016). Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(1), 142–158. https://doi.org/10.1109/TPAMI.2015.2437384

[14]: Girshick, R. (2015). *Fast R-CNN* (arXiv:1504.08083). arXiv. https://doi.org/10.48550/arXiv.1504.08083

[15]: Ren, S., He, K., Girshick, R., & Sun, J. (2016). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks* (arXiv:1506.01497). arXiv. https://doi.org/10.48550/arXiv.1506.01497

[16]: Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

[17]: Hui, J. (2019, April 12). Understanding Region-based Fully Convolutional Networks (R-FCN) for object detection. *Medium*. https://jonathan-hui.medium.com/understanding-region-based-fully-convolutional-networks-r-fcn-for-object-detection-828316f07c99

[18]: *Faster R-CNN: Down the rabbit hole of modern object detection*. (n.d.). Tryolabs. Retrieved June 15, 2024, from https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection

[19]: Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/CVPR.2016.91

[20]: *Darknet: Open Source Neural Networks in C*. (n.d.). Retrieved June 15, 2024, from https://pjreddie.com/darknet/

[21]: Redmon, J., & Farhadi, A. (2016). *YOLO9000: Better, Faster, Stronger* (arXiv:1612.08242). arXiv. http://arxiv.org/abs/1612.08242

[22]: Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). *SSD: Single Shot MultiBox Detector* (Vol. 9905, pp. 21–37). https://doi.org/10.1007/978-3-319-46448-0_2

[23]: Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement* (arXiv:1804.02767). arXiv. http://arxiv.org/abs/1804.02767

[24]: *YOLO Object Detection Explained: A Beginner's Guide | Encord*. (n.d.). Retrieved June 15, 2024, from https://encord.com/blog/yolo-object-detection-guide/

[25]: Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors* (arXiv:2207.02696). arXiv. https://doi.org/10.48550/arXiv.2207.02696

[26]: Priyanka Sisodia, S. B. (n.d.). An Implementation on Python for Data Science and Machine Learning. 10, 2320–2882.

[27]: Raschka, S., Patterson, J., & Nolet, C. (2020). Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information*, *11*(4), 193. https://doi.org/10.3390/info11040193

[28]: Sigut, J., Castro, M., Arnay, R., & Sigut, M. (2020). OpenCV Basics: A Mobile Application to Support the Teaching of Computer Vision Concepts. *IEEE Transactions on Education*, *63*(4), 328–335. https://doi.org/10.1109/TE.2020.2993013

[29]: Gangal, A., Kumar, P., & Kumari, S. (2021). *Complete Scanning Application Using OpenCv* (arXiv:2107.03700). arXiv. https://doi.org/10.48550/arXiv.2107.03700

[30]: Patnaik, R., Pradhan, R., Rath, S., Mishra, C., & Mohanty, L. (2021). Study on Google Firebase for Real-Time Web Messaging. In D. Mishra, R. Buyya, P. Mohapatra, & S. Patnaik (Eds.), *Intelligent and Cloud Computing* (pp. 461–469). Springer. https://doi.org/10.1007/978-981-15-5971-6_50

[31]: Moroney, L. (2017). An Introduction to Firebase. In L. Moroney (Ed.), *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform* (pp. 1–24). Apress. https://doi.org/10.1007/978-1-4842-2943-9_1

[32]: Pajankar, A. (2022). Introduction to Single-Board Computers and Raspberry Pi. In A. Pajankar (Ed.), *Raspberry Pi Image Processing Programming: With NumPy, SciPy, Matplotlib, and OpenCV* (pp. 1–30). Apress. https://doi.org/10.1007/978-1-4842-8270-0_1

[33]: Jolles, J. W. (2021). Broad-scale applications of the Raspberry Pi: A review and guide for biologists. *Methods in Ecology and Evolution*, *12*(9), 1562–1579. https://doi.org/10.1111/2041-210x.13652

[34]: creatorswarehouse. (2023, March 5). Orange Pi vs Raspberry Pi. Creators Warehouse. https://creatorswarehouse.com.au/orange-pi-raspberry-pi-comparison/

[35]: Tools, H. (2024, March 28). *Raspberry Pi 5 vs Orange Pi 5: An In-Depth Comparison for Makers and Hobbyists - History Tools*. https://www.historytools.org/vs/raspberry-pi-5-vs-orange-pi-5

[36]: Cass, S. (2020). Nvidia makes it easy to embed AI: The Jetson nano packs a lot of machine-learning power into DIY projects - [Hands on]. *IEEE Spectrum*, *57*(7), 14–16. https://doi.org/10.1109/MSPEC.2020.9126102

[37]: *OrangePi_5B_RK3588S_User Manual_v1.4.pdf*. (n.d.). Google Docs. Retrieved June 2, 2024, from https://drive.google.com/file/d/1n_sMewTbbszuEFHhpL-dNqHBiOwUtdJT/view?usp=sharing&usp=embed_facebook