People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University 8 May 45 – Guelma –
Faculty of Mathematics, Computer Science and Sciences of Matter
Department of Computer Science

Master Thesis

Specialty: Computer Science
Option:Science and Technology of Information and Communication

Theme

An Intelligent Vehicle Recognition System for Access Control in Algeria

---

Presented by: Balkiss Bouahia and Ikram Merour.

Jury Members

Chairman	Dr. Asma Boudria
Supervisor	Dr. Hassina Bouressace
Examiner	Pr. Chemesseen Nehar Bencheriet
Representative POLE-PRO	Dr. Zahra Abbas

June 2025

Acknowledgments

First and foremost, all praise is due to Allah (Alhamdulillah), who has made this journey possible. I am endlessly grateful for the opportunity, strength, and perseverance He has blessed me with to complete this work.

My deepest appreciation goes to my supervisor, **Dr. Bouressace Hassina**, for her unwavering support, invaluable advice, and expert guidance throughout this process. Her wisdom, patience, and encouragement have been instrumental in shaping this dissertation, and I am profoundly thankful for her mentorship.

To my beloved parents, **Dali Houria & Merour Ahmed**, I owe everything. Your unconditional love, endless prayers, and countless sacrifices have been my foundation. Your belief in me has carried me through every challenge, and I am forever indebted to your kindness and support.

My heartfelt thanks to my older brother, **Aymen**, for his constant motivation, protection, and the wisdom he has shared during difficult moments. Your presence in my life is a blessing I cherish deeply.

To my dear sister, **Rim Amani**, your love, gentleness, and unwavering encouragement have been a guiding light in my life. Thank you for always standing by me and believing in my dreams.

To my youngest brother, **Mohamed Iheb**, your innocence, laughter, and pure heart have brought me so much joy, especially during stressful times. You remind me daily of life's simple beauties.

A special thank you to my high school friend, **Roumaissa Amirouch**. Though we've taken different paths, the bond we share remains precious to me. I am grateful for the memories, laughter, and enduring friendship we've built over the years.

To my dearest university friends **"Zahra Aichour, Madjida Boularasse and Assala BenKamouch"** thank you for making my academic journey so meaningful. Your friendship has been one of my greatest treasures. From endless support to shared laughter and heartfelt conversations, every moment with you has been a gift.

I also extend my gratitude to my extended family and all those who have shown me kindness and encouragement along the way. Your love and belief in me have been a constant source of strength.

Last but not least, I wanna thank **me** for believing in me, for doing all the hard work, for never quite, for the late nights, and never giving up. Proud of how far I've come, and ready for what's next.

Dedications

Alhamdulillah ,All praise is due to Allah, the Most Merciful, the Most Wise.
Throughout this journey, one lesson I have truly learned in life is this:فَإِنَّ مَعَ الْعُسْرِ يُسْرًا
Indeed, with hardship comes ease.
This verse didn't merely echo in my mind, it anchored my soul. It reminded me that placing my trust in Allah is true strength and that behind every hardship lies a hidden mercy, and within every tear, a deeper purpose.And truly i have witnessed that ease.

I sincerely thank my supervisor, **Dr. Bouressace Hassina**, for her continuous support, valuable advice, and kind guidance. Her help was essential in completing this work, and I'm truly grateful for her mentorship.

To the roots of my strength and the warmth of my heart,**My parents**. my pillars in this life,I owe you everything. Your love is the kind that doesn't need to be spoken to be felt. Every silent prayer you made for me, every sacrifice you never mentioned.I carry them in my heart, forever. You have planted the seeds of patience, honesty, and strength in me, and today I bloom because of you.

To the two pieces of my heart, **Ilyess and Wassim**, my calm and my storm. **Ilyess**, your maturity, your advice, and the way you silently protect me mean more than you know. In your quiet strength, I find stability. **Wassim**, your laughter, innocence, and energy brought sunshine to my darkest days. You reminded me to breathe when I forgot how.

To my soulmate in friendship, my sister by heart, my teenage twin, **Roumi**.
Even when distance grew between us, your presence never left. You were my echo through the chaos, my reminder of who I was before the stress, before the race. Your words, your memories. You are proof that real connections don't fade — they evolve.

To my university companion **Madja Boularasse, Zahra Aichour, and Assala BenKamouch**.Thank you for the sweet days, the shared struggles, and the unforgettable moments.

This journey has shaped me. I've grown, I've learned, and I've changed.Today, I walk forward with strength in my heart, peace in my soul, and gratitude in every step. This is not just the end of a chapter, it's the start of who I am becoming.

Abstract

Vehicle recognition systems are essential tools in modern intelligent transportation and security infrastructures, enabling automated access control, traffic monitoring, and law enforcement. In this thesis, we propose an intelligent vehicle recognition system specifically designed for access control in Algeria. The system integrates several deep learning-based modules to identify and analyze key vehicle attributes, including license plate detection and recognition, vehicle orientation, color, brand, and model. The system handles both image and video inputs. To achieve this, we adopt the YOLOv8 model for multiple tasks, including vehicle detection, license plate localization, orientation estimation, and logo detection. The well-known OCR technique is also applied during the character recognition phase, while Convolutional Neural Networks are employed for vehicle color classification and logo recognition. Finally, MobileNet is used for vehicle model identification. Different datasets were applied, with a specific dataset used for each phase. The proposed multi-stage architecture achieves high accuracy, indicating strong performance and efficiency, which leads to a robust and intelligent vehicle recognition pipeline, offering a comprehensive solution that supports the core components of secure and automated access control systems.

Keywords: Vehicle Recognition, License Plate Detection, Character Recognition, Vehicle Orientation, Brand Recognition, Deep Learning, YOLOv8, CNN, MobileNet, Access Control, ALPR, Algeria.

Résumé

Les systèmes de reconnaissance de véhicules sont des outils essentiels dans les infrastructures modernes de transport intelligent et de sécurité, permettant le contrôle d'accès automatisé, la surveillance du trafic et l'application de la loi. Dans ce mémoire, nous proposons un système intelligent de reconnaissance de véhicules spécialement conçu pour le contrôle d'accès en Algérie. Le système intègre plusieurs modules basés sur l'apprentissage profond pour identifier et analyser les principaux attributs du véhicule, notamment la détection et la reconnaissance de la plaque d'immatriculation, l'orientation, la couleur, la marque et le modèle. Le système prend en charge les entrées sous forme d'images et de vidéos. Pour cela, nous adoptons le modèle YOLOv8 pour plusieurs tâches, notamment la détection de véhicules, la localisation des plaques d'immatriculation, l'estimation de l'orientation et la détection des logos. La technique bien connue OCR est également appliquée durant la phase de reconnaissance des caractères, tandis que les réseaux de neurones convolutifs sont utilisés pour la classification de la couleur des véhicules et la reconnaissance des logos. Enfin, MobileNet est utilisé pour l'identification du modèle de véhicule. Différents jeux de données ont été utilisés, chacun étant spécifiquement affecté à une phase donnée. L'architecture multi-étapes proposée atteint un haut niveau de précision, ce qui démontre sa performance et son efficacité, aboutissant à un système de reconnaissance de véhicules robuste et intelligent, offrant une solution complète répondant aux exigences d'un contrôle d'accès sécurisé et automatisé.

Mots-clés : Reconnaissance de véhicules, Détection de plaques d'immatriculation, Reconnaissance de caractères, Orientation du véhicule, Reconnaissance de marque, Apprentissage profond, YOLOv8, CNN, MobileNet, Contrôle d'accès, ALPR, Algérie.

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence

CNN Convolutional Neural Network

RCNN Region-based Convolutional Neural Network

SPP Spatial Pyramid Pooling

SPP-NET Spatial Pyramid Pooling Network

YOLO You Only Look Once

OCR Optical Character Recognition

NPR Number Plate Recognition

ANPR Automatic Number Plate Recognition

LPR License Plate Recognition

RPR Registration Plate Recognition

LLMS Large Language Model

LSTM Long Short-Term Memory

TROCR Transformer-based Optical Character

General Introduction

In today's world, the demand for automatic vehicle recognition systems is rapidly increasing due to their wide-ranging applications in security, law enforcement, traffic management, and smart city initiatives. These systems play a crucial role in enhancing public safety, streamlining vehicle identification, optimizing toll collection, and improving parking and access control. The advancements in deep learning and Optical Character Recognition (OCR) have significantly improved the accuracy and efficiency of such systems, making them indispensable in modern transportation infrastructure. Focusing on security surveillance, accurate vehicle recognition results are crucial to guarantee the system's efficiency.
In Algeria, vehicle recognition systems are still limited to a few developed implementations, where there is no advanced system that can specifically handle the complete range of Algerian vehicle characteristics, including license plates which have a different structure compared to other countries, with their own unique patterns. The Algerian plate basically combines numerical sequences with distinct formatting rules, color schemes, and font styles. These characteristics pose challenges for traditional vehicle recognition models, which are often trained on datasets that do not include a diverse range of Algerian vehicle features, where there are different types of distortions such as motion blur, focus blur, and poor lighting. In addition to that, the variations in plate size, font type, and potential future modifications make it more challenging, which necessitates a flexible and adaptable recognition system. These issues become even more challenging when dealing with video and live sequences in real scenarios of cars, trucks, etc.
In this Thesis, we focus on developing a robust and efficient vehicle recognition system, which is specifically developed for Algerian vehicles using deep learning and OCR, where the integration of OCR ensures precise extraction of alphanumeric characters from license plates, enabling reliable vehicle identification. Our system is designed to accurately detect, analyze, and recognize all vehicle characteristics found in Algeria, whether from videos or pictures, and is specifically designed for creating an advanced access control system.

This thesis is organized into four chapters, where each one focuses on key aspects of our developed system.

- Chapter 1 presents an overview of vehicle detection and recognition, highlighting its key components and the deep learning techniques involved. We provide an in-depth analysis of various vehicle detection algorithms, such as Faster R-CNN, YOLO, and SSD, emphasizing their strengths and limitations in relation to different parts of the vehicle.

- Chapter 2 is dedicated to exploring state-of-the-art techniques in vehicle identification, as well as license plate detection and recognition. We conduct a comprehensive

review of existing methods, including deep learning-based approaches, traditional computer vision techniques, and hybrid models.

- Chapter 3 focuses on the design of our system. In this chapter, we detail the initial stages of system development, including architectural choices and system planning.
- Chapter 4 concentrates on the implementation phase of the system development process. It introduces the working environment and presents the results obtained from evaluating the performance of the implemented algorithms.

Chapter 1

Vehicle Recognition and Deep Learning

1.1 Introduction

Deep learning has transformed the field of computer vision by enabling the automatic extraction of complex features directly from raw image data. Many deep learning models have achieved remarkable success across a wide range of computer vision tasks, particularly in vehicle recognition. Their ability to automatically learn hierarchical features from images has made them indispensable in applications requiring precise and efficient visual recognition. This chapter provides an in-depth overview of vehicle recognition methods and their impact on advancing the field. It also presents access control systems developed based on vehicle recognition results, along with specifying the features of Algerian license plates.

1.2 Vehicle Identity Recognition

Several phases and techniques are crucial for an effective and accurate access control system that covers both visual identity and functional classification of the vehicle.

1.2.1 License Plate Detection and Recognition

The license plate, often known as the registration number, is a metal plate that is affixed to the front and rear of a vehicle. It makes it possible to identify the car specifically. The combination of letters and numbers that the appropriate government assigns to the vehicle makes up the registration number. The license plate is essential for managing and regulating traffic as well as for upholding rules pertaining to automobiles and traffic [1]. A license plate is the unique identity of the vehicle, which serves as proof of the legitimacy of the operation of the vehicle in the form of a plate or other material with certain specifications issued by the police [2].(see Figure 1.1)

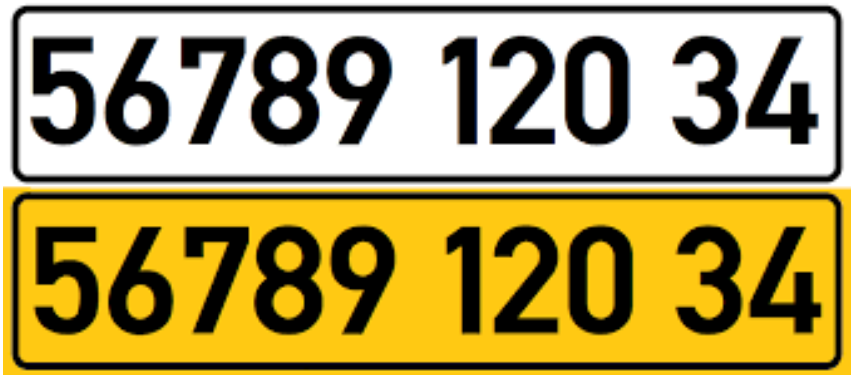


Figure 1.1: Example of a general structure of a vehicle license plate

License Plate Detection refers to the process of automatically locating the position of the license plate within a digital image or video frame. This is typically the first step in any license plate recognition system. It involves identifying and isolating the rectangular area in which the plate is placed, regardless of lighting, orientation, or background noise. as shown in Figure 1.2.

Figure 1.2: Example of License Plate Detection

License Plate Recognition, on the other hand, involves extracting the alphanumeric characters from the detected plate area and converting them into readable and searchable text. This phase may require pre-processing steps such as resizing, binarization, character segmentation, and classification using machine learning or OCR (Optical Character Recognition) techniques. as shown in Figure 1.3

Figure 1.3: Illustration of The License Plate Recognition Process

1.2.2 Vehicle Visual Identity Techniques

In addition to the license plate, the visual features of the vehicle such as color, brand, and type play a major role in confirming or cross-verifying the identity of a vehicle. These methods are commonly used in intelligent transportation systems, access control, and security applications.

A. Color Detection and Recognition

The color of a vehicle is a crucial characteristic for identification and for providing access control with visual indications for prompt action. Vehicle color recognition is an extremely difficult assignment due to a number of elements, such as the weather and the quality of the video or image capture [3]. As shown in Figure 1.4.

Figure 1.4: Example of Color Detection and Recognition[4].

- **Common Techniques :**

  1. **Color Space Conversion (e.g., RGB to HSV or LAB):**
     Converting RGB images to other color spaces like HSV helps isolate hue (color tone) from lighting, making detection more reliable. [5]
  2. **Clustering (e.g., K-means):**
     K-means is used to group similar colors and extract the most dominant ones from an image.[6]
  3. **Deep Learning Models (e.g., CNN):**
     CNNs can be trained to recognize colors as part of a broader classification system (e.g., car color classifiers).[7]

B. Vehicle Brand/Model Recognition

The process of determining a vehicle's manufacturer (make) and particular model (the product the manufacturer sells to customers) from pictures or videos is known as Vehicle Make and Model Recognition (VMMR)[8]. As shown in Figure 1.5.

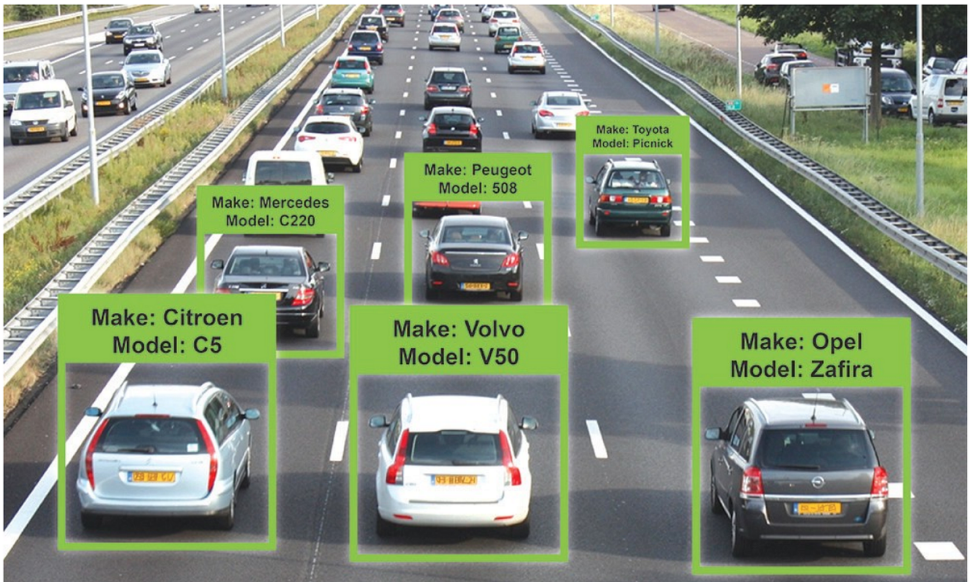


Figure 1.5: Example of Brand and Model Recognition[9].

- **Common Techniques :**

  1. **Feature-Based Methods**
     Feature-based approaches, such as HOG and SURF, employ car photos to pinpoint particular makes and models. Vehicles are categorized using classifiers such as SVMs using features that have been retrieved, however illumination might affect performance. [10]
  2. **Deep Learning-Based Methods**
     Deep learning techniques, particularly convolutional neural networks, have revolutionized VMMR by learning hierarchical features from image data, capturing complex patterns, and enhancing accuracy with multi-task learning frameworks and attention mechanisms. [8]

C. Vehicle Type/Category Recognition

Refers to the process of automatically identifying the general class to which a vehicle belongs, based on visual features extracted from images or videos.
A commonly used classification includes the following categories:[11]

- **1** : Passenger vehicles
- **2** : Trucks
- **3** : Vans
- **4** : Buses and coaches
- **5** : Road tractors
- **6** : Other tractors
- **7** : Special vehicles

- **8** : Trailers and semi-trailers
- **9** : Motorcycles

D. Vehicle Side/Orientation Detection

The term "vehicle orientation" refers to the angle or direction from which a vehicle is viewed in an image or video. (e.g., front, rear, side) helps in determining which features to extract and supports other recognition tasks [12]. As shown in Figure 1.6 that represents the result of orientation detection.

Figure 1.6: Example of Vehicle Side Detection[13].

- **Common Techniques :**
  1. Convolutional Neural Networks (CNNs) trained to classify view angles. [14]
  2. YOLO or other object detectors trained with orientation labels. [15]
  3. Hybrid methods using part detection (e.g., headlights, taillights) to infer side

1.3 Models of Plate Detection

Plate detection models are designed to locate and extract license plates from images or videos. They utilize machine learning and deep learning techniques, such as CNN and object detection algorithms, to ensure accuracy and efficiency in diverse environments.

1.3.1 Region-based Convolutional Neural Networks(R-CNN)

A class of machine learning models called region-based Convolutional neural networks (R-CNN) is intended for computer vision applications such as object detection and localization. R-CNN, which was first presented by Ross Girshick and associates in 2014,

uses region suggestions in conjunction with Convolutional Neural Networks (CNN) to recognize and categorize objects in a picture [16]. As shown in Figure 1.7

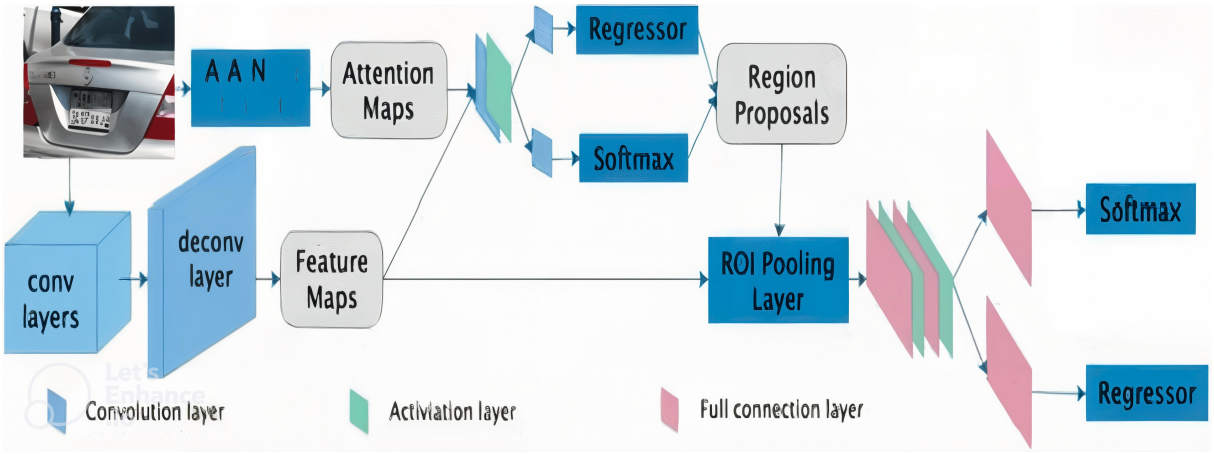


Figure 1.7: Architecture of RCNN.[16]

To increase localization accuracy, the system uses selective search to produce region suggestions, extract feature vectors, classify objects using SVMs, and modify bounding boxes using a regression model. Each proposal must be resized, run through a CNN that has already been trained, and then resized. [17].
R-CNN is highly effective for license plate detection (a critical task in ALPR systems) due to its ability to localize and classify irregularly shaped object

- **Steps for Plate Detection Using R-CNN:**

  1. **Region Proposal Generation:**
     Selective search is used to generate region proposals (candidate regions) in the input image where license plates might be located; this step avoids exhaustive search by proposing regions likely to contain objects (plates).
  2. **Feature Extraction :**
     Each suggested region is run through a CNN that has already been trained (e.g., AlexNet, VGG) and warped or scaled to a specified size (e.g., 227x227 pixels). Each region's high-level features, such as edges, textures, and characters, are extracted by the CNN.
  3. **Classification (Softmax/SVM):**
     To ascertain whether or not the area has a license plate, extracted characteristics are passed into a classifier (initially SVM in R-CNN, subsequently replaced by Softmax in Faster R-CNN). Classification in binary: "Plate" vs "Non-Plate."
  4. **Bounding Box Regression :**
     The suggested bounding boxes' coordinates are fine-tuned by a regression model to closely match the identified license plates. This modifies the box's size and position to increase localization accuracy.
  5. **Post-Processing (Optional):**
     Only the most certain plate regions remain after Non-Maximum Suppression (NMS) eliminates redundant or overlapping detections.

1.3.2 Spatial Pyramid Pooling Network(SPP-NET)

An extension of convolutional neural networks (CNNs), the Spatial Pyramid Pooling Network (SPP-NET) adds a Spatial Pyramid Pooling (SPP) layer between the last convolutional layer and the fully connected layers. It is no longer necessary to resize or crop input photos because of this design, which enables the network to receive input images of different sizes and produce fixed-length feature representations.[18] as shown in Figure 1.8

This is crucial for license plate detection because, in real-world situations, plates may appear at various scales or aspect ratios (e.g., near/far vehicles).[19]

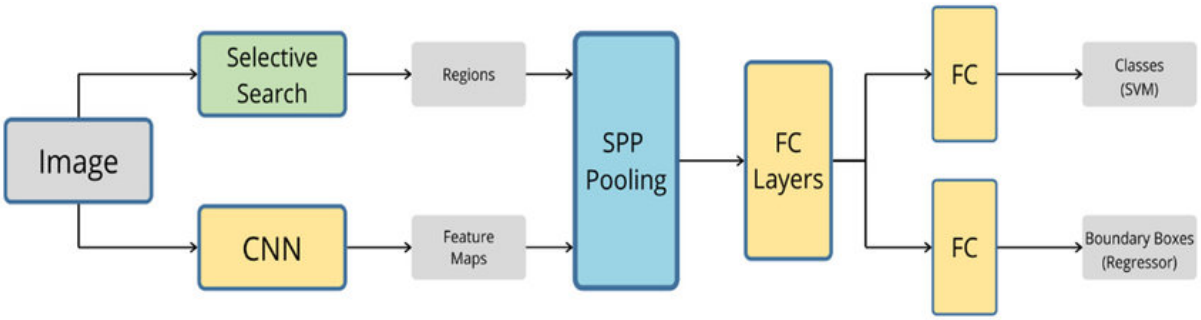


Figure 1.8: Architecture of SPP-net.[20]

- **Key Steps for Plate Detection:**

  1. **Input Image & Region Proposals:**
     Selective Search generates candidate regions (bounding boxes) likely to contain plates, preserving aspect ratios. And its advantage that it can avoids distortion from fixed-size warping (unlike R-CNN) [21]
  2. **Feature Extraction with SPP Layer:**
     - **The CNN backbone**: (e.g., ZFNet, VGG) processes the entire image once
     - **SPP Layer**: Divides region proposals into spatial bins, pooling features to create a fixed-length vector, maintaining plate features regardless of scale/skew.[19]
  3. **Classification & Localization:**
     - **FC Layers** : SPP outputs should be flattened for categorization.
     - **SVM/Softmax** :Classifies regions as "Plate" or "Non-Plate" (originally SVM).[21]
     - **Bounding Box Regression** : For accurate plate localization, region coordinates are adjusted (e.g., correcting slant). [19]
  4. **Output:**
     High-confidence detections after Non-Maximum Suppression (NMS).

1.3.3 Only Look Once (YOLO)

One popular and viral algorithm is You Only Look Once (YOLO). YOLO is well-known for its ability to detect objects. The first YOLO version was introduced by Redmon and al. in 2015. Several versions, including YOLOv2, YOLOv3, YOLOv4, and YOLOv5, have been released by researchers in recent years. Additionally, some lightweight and optimized versions, such as YOLO-LITE, have been developed. This study focuses only on the five primary YOLO versions [22]. as shown in Figure 1.9

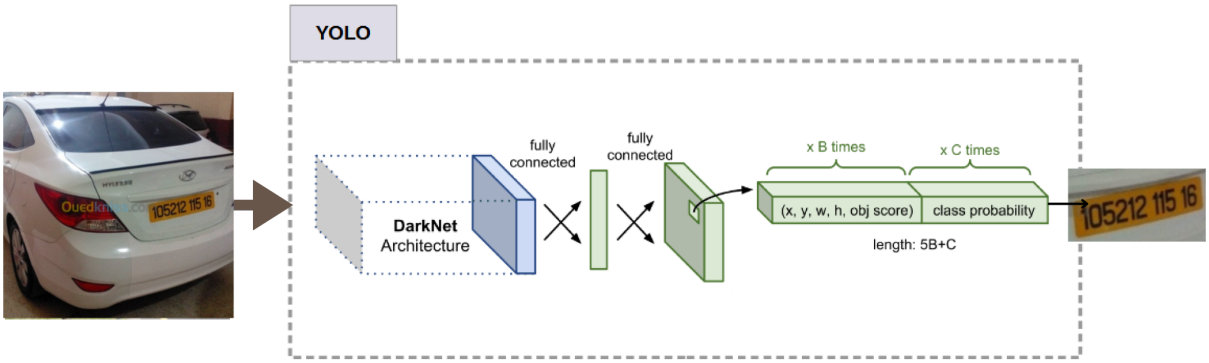


Figure 1.9: Architecture of YOLO.[23]

YOLO splits the input image into a grid for license plate detection and makes direct predictions about bounding boxes, confidence scores, and class probabilities.

YOLO (You Only Look Once) performs real-time license plate detection by dividing the input image into an S×S grid, where each grid cell predicts B bounding boxes (with coordinates x, y, w, h), an abjectness score (confidence the box contains a plate), and C class probabilities (e.g., "Plate" vs. "Non-Plate")[24].

The model processes the entire image in a single forward pass through a DarkNet CNN backbone, extracting features to predict these values simultaneously, followed by Non-Maximum Suppression (NMS) to eliminate redundant detections [25].

His unified approach enables high-speed processing (e.g., 45 FPS in YOLOv3) while maintaining accuracy across varying plate sizes and orientations. [26]

1.4 Models for License Plate Character Recognition

The second phase following plate detection is license plate recognition, which consists of multiple components. This improved computer vision technology phase, known as vehicle number plate recognition (NPR), license plate recognition (LPR), or registration plate recognition (RPR) links automobiles. without establishing a direct personal connection via their license plates. [27].

1.4.1 Convolutional Neural Network (CNN)

CNNs are primarily used in the initial stage of the license plate recognition process to extract key features from the detected license plate. This feature extraction step helps identify important patterns, such as shapes, edges, and text regions, which are essential

for recognizing the license plate characters (numbers and letters). CNNs are particularly effective in handling a diverse range of environmental conditions, including motion blur, poor lighting, and varying plate types, making them crucial for accurate recognition.

Inspired by the structure of the animal visual cortex, CNN is a kind of deep learning model for processing data with a grid pattern, such as photographs. It is intended to learn spatial feature hierarchies from low- to high-level patterns automatically and adaptively. Convolution, pooling, and fully connected layers are the three types of layers (or building blocks) that make up a CNN, a mathematical construct[28].
To increase localization accuracy, the system uses selective search to produce region suggestions, extract feature vectors, classify objects using SVMs, and modify the bounding boxes using a regression model. Each proposal must be resized, run through a CNN that has already been trained, and then resized[17]. Figure 1.10 and presents the basic steps of this model.

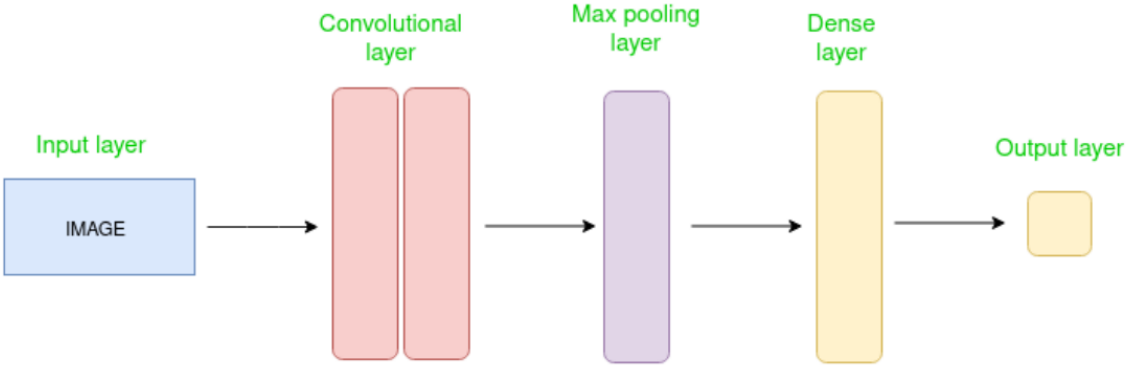


Figure 1.10: Basic convolutional neural network architecture[1].

- **Convolutional layer**

  In a CNN, a convolutional layer applies a set of learnable filters (also known as kernels) to the input data. As these filters slide (convolve) over the input's spatial dimensions, they compute dot products to produce feature maps. This process enables the network to detect various features, such as edges, textures, or patterns in the data. [29].

- **pooling layers**

  Pooling layers, also known as downsampling, reduce dimensionality, reducing the number of parameters in the input. Similarly to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array. There are two main types of pooling: [30].

  1. **Average Pooling**
     As the filter moves across the input, it calculates the average value within the receptive field to send to the output array. [30].

2. **Max Pooling**
   As the filter moves across the input, it selects the pixel with the maximum value to send to the output array.Also, this approach tends to be used more often compared to average pooling.[30].

- **Dense layer**

  The name of the fully connected layer aptly describes itself. As mentioned earlier, the pixel values of the input image are not directly connected to the output layer in partially connected layers. However, in the fully connected layer, each node in the output layer connects directly to a node in the previous layer.[30].

1.4.2 Recurrent Neural Network (RNN)

RNNs are primarily used as the next stage in the license plate recognition process, where they are responsible for recognizing the sequence of characters (numbers and letters) extracted from the license plate image. Unlike CNNs, which focus on extracting spatial features, RNNs handle the sequential nature of the plate characters, making them well-suited for recognizing text in varying lengths and formats. RNNs are particularly effective in processing sequences, allowing them to capture dependencies between adjacent characters, which is essential for accurate character recognition. When combined with CNNs for feature extraction, RNNs help decode the characters on the license plate, even under challenging conditions such as distorted or occluded text.

A recurrent neural network (RNN) is a deep learning model that is trained to process and convert a sequential data input into a specific sequential data output. Sequential data are data, such as words, phrases, or time series data, in which the sequential components are interconnected based on complex semantics and syntax rules. An RNN is a software system composed of numerous interconnected components that mimic the way humans perform sequential data conversions, such as translating text from one language to another.RNNs are largely being replaced by transformer-based artificial intelligence (AI) and large language models (LLMS), which are much more efficient in sequential data processing.[31]
Figure 1.11presents the basic steps of this model.

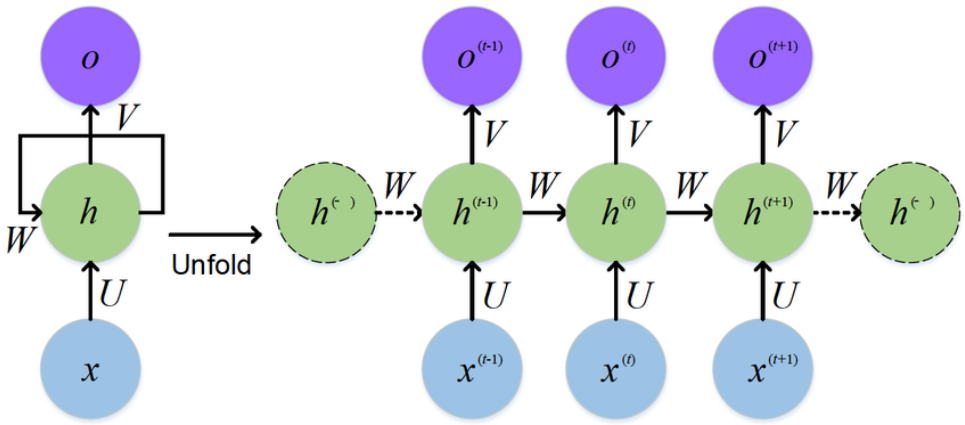


Figure 1.11: Basic recurrent neural network architecture.[32]

1.4.3 Optical Character Recognition (OCR)

Recent advances in OCR leverage deep learning techniques, such as CNNs to improve accuracy and robustness against noise, distortions, and varying fonts or handwriting styles, this makes OCR as an essential component in many applications, including document digitization, assistive technologies for visually impaired individuals, and in our case automated number plate recognition.
Optical Character Recognition (OCR) is a technology that converts images that comes in many types; handwritten and printed into editable text data, where its process involves several steps; preprocessing for enhance quality purposes, segmenting the text into individual characters or words, and then recognizing these segmented characters using pattern recognition algorithms. OCR systems main objective is to take segmented character images as input which is the previous phase's output and classify each character based on learned features, this enables the automatic transcription of textual content from various sources such as scanned documents, photographs, or license plates [33]. Figure 1.12 presents the basic steps of the OCR technique as applied to the vehicle recognition domain, specifically focusing on license plate recognition. In this process, the input is an image, and the output is a digital representation of the characters (usually numbers) on the license plate.

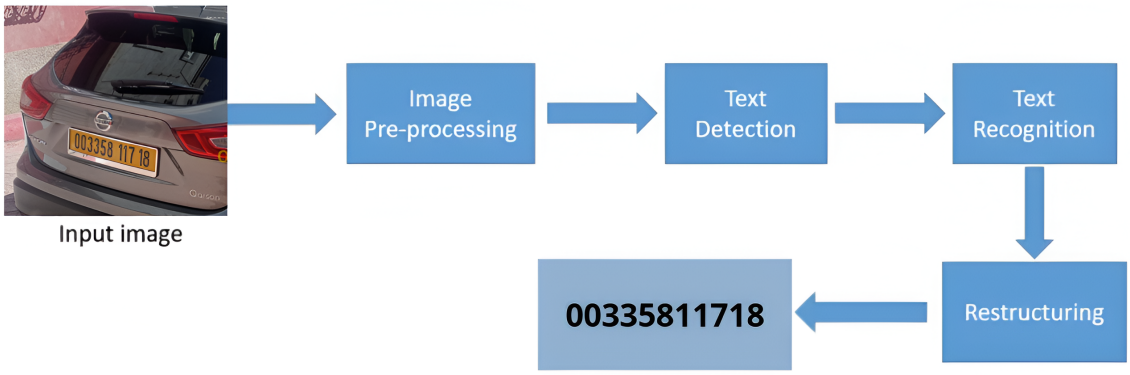


Figure 1.12: Basic Optical Character Recognition architecture.[34]

- **Input Image**
  A raw image of a vehicle, typically captured by a camera, serves as the input to the system. This image includes the license plate that will be processed for character extraction.

- **Image Pre-processing**
  Enhances image quality using techniques like resizing, grayscale conversion, and noise reduction. This prepares the image for more accurate text/plate detection.

- **Text Detection**
  Identifies the region of interest (the license plate) within the image using detection algorithms. This step isolates the plate from the rest of the image.

- **Text Recognition**
  Applies OCR or a character recognition model to the detected plate region. and Converts visual characters into digital text.

- **Restructuring**
  Reformats or corrects the recognized text to match the expected license plate format, This ensures uniformity for storage, analysis, or comparison.

- **Final Output**
  Displays or stores the cleaned and structured license plate number. This is the result of the end-to-end detection and recognition pipeline.

1.5 Applications of License Plate and Vehicle Recognition in Access Control

License plate and vehicle detection and recognition play a key role in traffic management and security. This application automates license plate identification and vehicle characteristics using computer vision, enhancing efficiency and accuracy while reducing manual effort. Some of these applications are:

1.5.1 Access Control and Parking Management

In parking lots and secured areas, license plate recognition systems are used to automatically control vehicle access and exit. These systems can improve security, expedite parking operations, and give or prohibit access by identifying license Plates [35]. (see Figure 1.13)

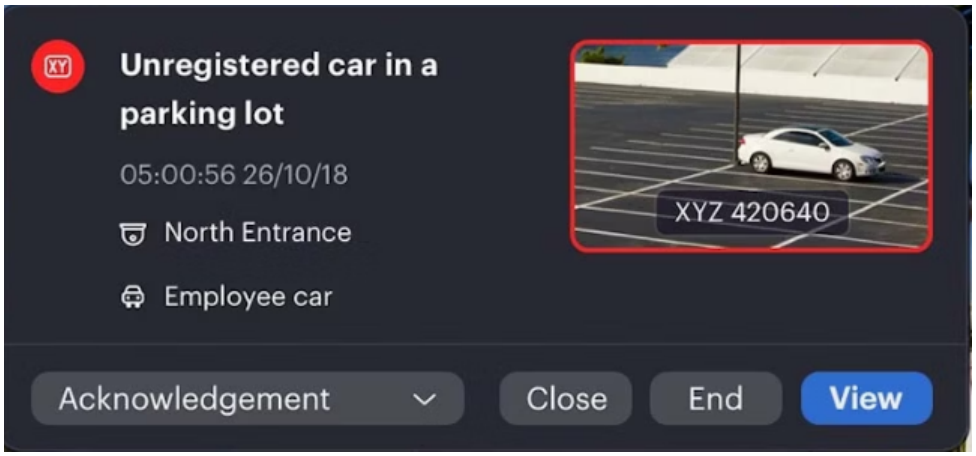


Figure 1.13: Example of Parking Management.[35]

1.5.2 Integration with Boom Barrier for Automated Access Control

Reliance on human barrier operations is decreased by a smooth integration with boom barriers. In addition to ensuring that only allowed vehicles enter the site, it can identify license plates recorded in a database and will sound an alert if any unauthorized plates are detected. As a result, Automatic Number Plate Recognition (ANPR) makes it possible to manage toll booths effectively, reducing the amount of time required for operations and boosting output[36].

This facilitates contactless access, and eliminating the need for physical tickets or access cards, which not only improves convenience but also reduces operational costs and potential security breaches.

These systems allow real-time monitoring, data logging, and reporting of entry and exit events, which also enhances the efficiency of high-traffic environments such as university campuses, airports, and shopping centers by reducing congestion at entry points. Studies show that when combined with vehicle recognition, these systems can also track vehicle types and detect unauthorized vehicles based on predefined access rules[37]. Figure 1.14 shows an example of parking management with an access control system that identifies accepted vehicles allowed to enter and those that are not.

Figure 1.14: Example of Integration with Boom Barrier.[36]

1.5.3 Blacklisted Vehicle Detection

Blacklisted vehicles can be automatically identified, and real-time notifications are delivered so that urgent action can be taken. Finding out if a car is registered and identifying vehicles involved in traffic violations are only two of the many uses for ANPR. Through real-time number plate detection and recognition, law enforcement ANPR technology makes sure that only authorized vehicles occupy parking spaces[36]. (see figure 1.15)

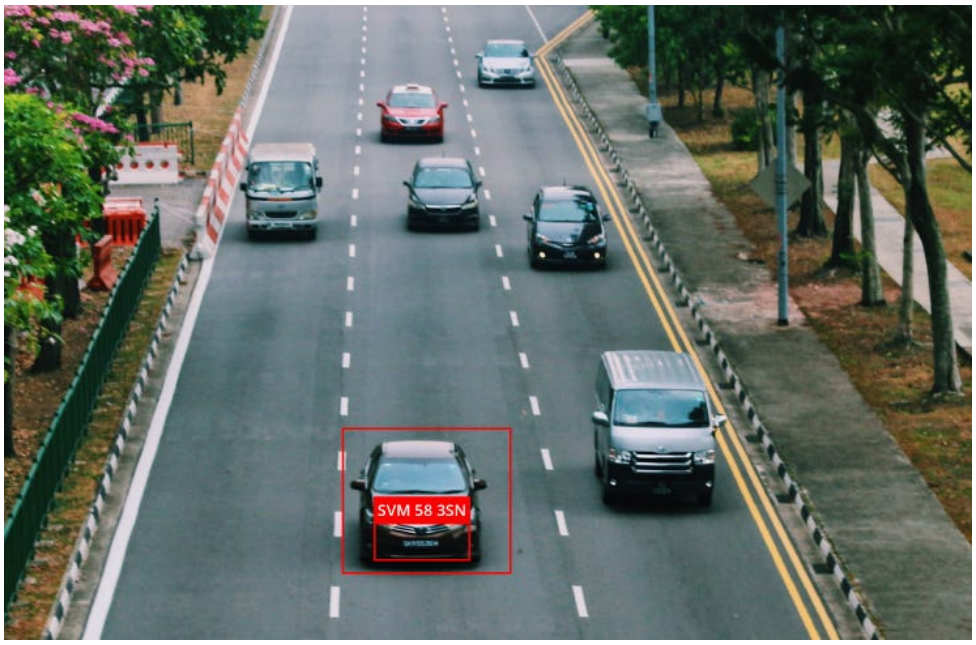


Figure 1.15: Example of Blacklisted Vehicle Detection.[36]

1.6 Algerian License Plates

The Algerian license plates are characterized by their unique features. In this section, we will present these features, which are divided into structure, characteristics, and the encountered and existing problems with these plates

1.6.1 Plate Structure

Three sets of numerals, positioned from right to left and separated by dashes, make up Algerian license plates, as shown in figure 1.16

1. **Wilaya Code (1-58)**: Denotes the province of registration.
2. **Three-digit vehicle information**:
   - Year of manufacturing is the first two digits.
   - Vehicle category is the third digit.
3. **Registration number**: The wilaya ranking for the registration year.[11]

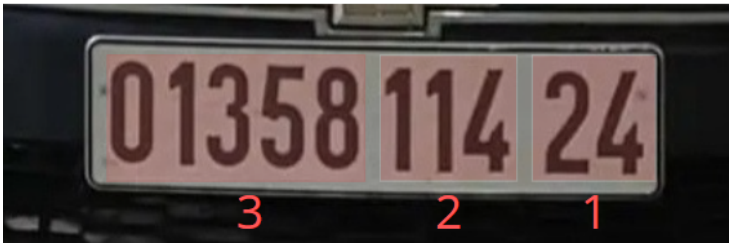


Figure 1.16: Example of structure of Algerian plates

1.6.2 Plate Characteristics

The majority of Algerian license plates are made of aluminum and have specific design features.The front plate displays Arabic numerals on a gray-white reflective background, while the rear plate has black Arabic numerals on a yellow reflective background. Temporary circulation plates use black Arabic numerals on a white reflective background for both the front and rear. The plates follow strict dimensions: 1 mm thickness, 52 cm length, 11 cm width, with numbers 7.5 cm high and spaced 0.5 mm apart.
(see figure 1.17) [11]

Figure 1.17: Example of Algerian plates[11].

While most Algerian license plates follow these features, there are other types of plates that do not conform to these characteristics, such as those used for specific purposes like diplomatic vehicles, temporary registrations, or certain government-issued plates. Figure 1.18presents a different Algerian plate model.

Figure 1.18: Algerian army registration plate[38].

1.6.3 Problems and Challenges

There are several problems involved in automatic license plate detection and recognition. Here we have explored few major problems:

1. **License Plate Non-Uniformity**
   the variations in license plate models throughout several cities. Additionally, it could differ from state to state and, consequently, from car to car. The number plates' lengths can also change.[39]

(a) (b)

Figure 1.19: Example of Algerian plates.[40]

2. **Blurred Plate Issue**
   the low quality of car license plates in video frames captured by standard camera systems. [39]

Figure 1.20: Example of Blurred Plate.[40]

1.7 Conclusion

In conclusion, license plate and vehicle detection and recognition is a crucial field that requires further development and expansion to enhance traffic monitoring, security, and automated vehicle identification. Advanced deep learning techniques, such as RCNN, YOLO, CNN, and OCR, offer viable solutions for improving accuracy and efficiency in automatic license plate recognition (ALPR) systems. The next chapter will examine current studies aimed at license plate and vehicle detection and recognition for access control systems.

Chapter 2

State-of-the-Art

2.1 Introduction

Automated Vehicle Recognition (AVR) is a technology that automatically detects, identifies, and classifies vehicles from images or video streams. It leverages computer vision, image processing, and deep learning techniques to recognize various visual attributes of vehicles, such as color, make and model, type or category, and orientation. AVR has become a key innovation in the transportation and security sectors, enabling enhanced applications in toll collection, intelligent parking systems, smart surveillance, and traffic flow analysis. This chapter provides an overview of the techniques used for vehicle detection and recognition, reviewing both traditional image processing approaches and recent advances in deep learning. The integration of these modern techniques has significantly boosted the accuracy, robustness, and applicability of AVR systems in real-world scenarios.

2.2 Automatic Vehicle Recognition and classification

This section covers the latest advancements in automatic vehicle recognition, focusing on techniques used to identify and classify vehicles. It includes: make and model recognition, color detection, orientation and side detection, vehicle type detection, and integration of the recognition techniques for these classifications.

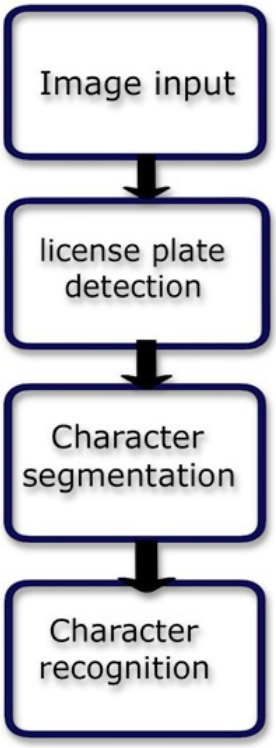


Figure 2.1: Basic organization of an ALPR system.[41]

2.3 License Plate Detection

Accurate detection of the license plate is crucial for the success of an ALPR system. Several methods have been proposed for this task.

2.3.1 Traditional Image Processing Techniques

A. Edge Detection

Recent studies have explored various methods to improve license plate detection using edge detection techniques.

The authors in [42] for edge detection algorithm combined with smart exposure to extract the license plate region. This method aims to overcome the limitations of traditional algorithms, such as Sobel, which may be ineffective under intense lighting conditions. By adjusting image brightness and reducing noise, the algorithm enhances the accuracy of plate detection even in challenging lighting environments.

Another study in [43] presents a two-step approach for license plate detection. The first step uses the Faster R-CNN model to locate the plate. This method has shown significant performance improvements, achieving a recall rate of 92% on the CENPARMI dataset and 90% on the UFPR-ALPR dataset.

Additionally, a recent study explored the use of computer vision techniques to develop an efficient vehicle number recognition system, facilitating automated toll collection. This system identifies the vehicle type, captures a front-facing image of the vehicle, locates and segments the license plate characters, and recognizes them to extract the license plate number. [44]

These studies highlight recent advancements in license plate detection, emphasizing the effectiveness of edge detection techniques and deep learning-based approaches to enhance the accuracy and robustness of license plate recognition systems under various conditions.

B. Texture-Based Methods

In [45], a license plate detection method based on the Wavelet transform is described. This approach utilizes the HL sub-band of the Wavelet transform for feature extraction, while the HL sub-band is used to verify the presence of a horizontal line around the detected feature. The authors report a localization accuracy of 97.33% using this method.

One of the advantages of texture-based methods, such as the one described in [45], is their robustness against license plate deformation. This means that even if a license plate is bent or damaged due to various factors (wear and tear, accidents, etc.), the method remains effective in detecting and recognizing it.

However, these approaches also have limitations. They often require complex computations and can be sensitive to cluttered backgrounds or varying lighting conditions. For instance, in an environment with a busy background or poor illumination, extracting texture features may be challenging, which could compromise license plate detection. As a result, these methods are not always suitable for all scenarios and may require adjustments or optimizations to enhance their performance based on the context.

In [46], a scan-line technique was employed for license plate detection. This method is based on the observation that the region containing the license plate exhibits a unique level of complexity in a grayscale image, making it a distinctive feature that can be leveraged for detection. Unlike approaches that rely on plate boundary details, this method is more robust and can be applied in various scenarios.

Zunino and al. [47] introduced an innovative localization method based on Vector Quantization (VQ). Unlike techniques that focus solely on features such as edges and contrast, this approach considers the actual content of the license plate. The authors reported a detection accuracy of 98%, with the method being tested in a real-time industrial application.

License plates introduce inconsistencies in the texture of an input image, which can reveal their presence. The unique texture and appearance of the letters and numbers on a license plate allow them to be distinguished from the surrounding background. By utilizing texture-based approaches like Vector Quantization, these unique features can be leveraged to accurately detect and recognize license plates in various conditions. However, as mentioned earlier, these methods may not be suitable for all scenarios and often require further optimization to improve their robustness.

2.3.2 Deep Learning-Based Techniques

A. Convolutional Neural Network (CNN)

The creators of [48] introduced a framework utilizing convolutional neural networks (CNNs) to identify vehicle license plates. Their method enhances existing techniques for recognizing blurry and obscured images and operates efficiently across different lighting scenarios.

CNNs represent a category of neural networks frequently applied in computer vision tasks, such as object detection and image classification. They excel at handling extensive image datasets, making them particularly suitable for detecting license plates. The authors of [48] trained their CNN using a substantial collection of license plate images to identify the crucial features for plate detection. By exposing the network to a diverse array of images, the CNN has been trained to recognize license plates in various lighting conditions and settings.

The accuracy achieved by their proposed method is reported to be approximately 100%. This suggests that their system can accurately identify license plates in every instance, attaining a flawless score within their testing dataset. Nonetheless, it is essential to recognize that the accuracy observed in a test dataset might not always reflect real-world effectiveness. In practical applications, there could be additional challenges, such as fluctuations in lighting and environmental factors, which might impact the system's accuracy.

In summary, the CNN-based framework presented by the authors of [48] provides a promising approach for license plate detection. By employing a neural network methodology, their system successfully learns to identify license plates across a broad range of circumstances and demonstrates high accuracy on their testing dataset.

In the research conducted by Selmi and al.[49], a localization method based on CNNs was proposed, comprising two key phases: preprocessing and classification. During the preprocessing phase, the input image was refined to eliminate noise and capture finer details. The classification phase utilized a CNN classifier to differentiate potential bounding boxes into either license plate or non-plate areas. This approach was tested on the Caltech dataset, yielding a recall accuracy of 93.8% and an F-score accuracy of 91.3%.

Another research effort by Zou and al.[50] involved training two CNN models—shallow CNN and deep CNN—from end to end for license plate detection. The shallow CNN aimed to lower computational demands by filtering out most background areas from the image, whereas the deep CNN focused on detecting the license plate from the remaining sections. Non-maximum suppression (NMS) was subsequently employed to pinpoint the precise plate location. The experimental findings indicated above-average accuracy with reduced computational expense. Overall, these studies highlight the effectiveness of deep learning neural networks, especially CNNs, in the realm of license plate detection. These methods have shown encouraging outcomes and possess potential for further enhancement in practical applications.

B. You Only Look Once (YOLO)

The You-Only-Look-Once (YOLO) framework has become a cornerstone in real-time object detection, including license plate recognition, due to its speed and accuracy.

In [51], Silva and al. utilized YOLOv2 to detect vehicles without modifications, treating the network as a black box and merging car and bus classes from the PASCAL-VOC

dataset. This approach demonstrated the adaptability of YOLO for vehicle detection tasks. Meanwhile, WPOD-NET, inspired by YOLO, SSD (Single Shot Detector), and STN (Spatial Transformer Network), was specifically designed to handle distorted license plates by regressing affine transformation coefficients to unwrap plates into a frontal view, improving detection accuracy.

In order to precisely locate the license plate, Min and al[52] suggested a solution that uses the most recent YOLO-L model and pre-identification plate. There are two components to the suggested approach. First, the right number and size of candidate boxes for the license plate were chosen using the k-means++ clustering technique. The YOLOv2 network model and depth were then adjusted to precisely detect the license plate. Additionally, to distinguish license plates from other objects, a pre-identification method was used. When tested on a dataset of various license plate kinds, the suggested approach produced 98.86% precision and recall. The efficiency of using the YOLO-L model and pre-identification algorithm for license plate detection is demonstrated by the high accuracy attained by the suggested method. Applications for this system include police enforcement, traffic management, and the Parking toll collection system.

Additionally, Xie and al. introduced MDYOLO in [53], an enhanced YOLO framework capable of detecting multi-directional license plates by predicting rotation angles. By redesigning the CNN architecture and introducing a novel loss function, MDYOLO outperformed traditional YOLO in accuracy and robustness, particularly in challenging lighting and occlusion conditions. These advancements highlight the flexibility and power of YOLO-based methods, which continue to evolve with new techniques like affine transformations, pre-identification algorithms, and multi-directional detection, making them highly effective for license plate recognition in diverse scenarios.The figure 2.2 below shows the redesigned CNN architecture for the MDYOLO model.

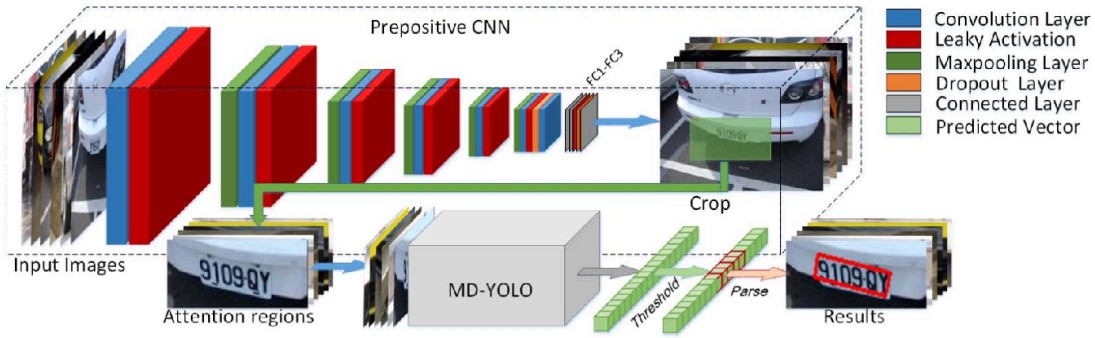


Figure 2.2: CNN architecture for the MD-YOLO model.[51]

2.4 Character Segmentation

In the character segmentation phase, many techniques have been developed to improve accuracy and efficiency in the results. In this part, we will focus on the most well-known ones that have been used by researchers in recent years.

In [54], the CCA (Connected Component Analysis) scans and classifies the pixels of a binarized image into components based on pixel connection. Each pixel has a value

assigned to it based on the component to which it was allocated. After that, the linked components are examined to eliminate lengthy and wide components, leaving only those that meet the predetermined criteria. However, the disadvantage of this approach is that the segmented result might not include the precise license plate locations.

In [55], this algorithm overcomes the limitations of previous methods like vertical projection and connected component analysis (CCA) by adopting Niblack's method as the binarization algorithm. It consists of three sub-parts: adjusting the input image, generating a binary image, analyzing blobs, and selecting the final seven character blobs. The method achieves a 97.2% success rate when analyzing failure cases closely, but has a single parameter for Niblack's binarization method, which may not be optimal for each image.

In [56], the preprocessing stage of character segmentation is enhanced with a projection-based method, consisting of coarse segmentation and fine segmentation. Coarse segmentation involves vertical projection to the preprocessed plate, obtaining nonzero blocks as candidate characters. Horizontal projection is applied to each character, resulting in a common bounding box. Fine segmentation addresses three major problems: coarse segmentation is not ideal, broken characters require merging and splitting, and vertical frames may connect with first and last characters. This method improves character recognition accuracy by combining projection and character characteristics, achieving 97% accuracy. However, it often results in errors in fuzzy images.

2.5 License Plate Recognition

2.5.1 Convolutional Neural Network (CNN)

In the study referenced in [57],Alam developed a method for identifying Bengali language vehicle number plates using Convolutional Neural Network (CNN) and Deep Learning strategies. The system aims to accurately store vehicle numbers in a cloud-based system. The technique uses a super-resolution technique to reconstruct the pixel quality of the input image, improving its accuracy. The number plate recognition system uses bounding box techniques to segment each character, allowing accurate identification. Alam tested the system on 700 vehicles, achieving 98.2% accuracy in the validation set and 98.1% in the evaluation set, with an error rate of 1.8%.

In the study referenced in[58], Shaifur Rahman and al developed a Convolutional Neural Networks (CNN)-based Bangla license plate recognition system (BLPRS) for efficient roadside assistance and vehicle license status identification, aiming to improve vehicle identification. The authors trained a BLPRS using six CNN layers and a fully connected layer to accurately recognize license plate numbers in license plate images. They tested the system on a testing dataset, achieving 89% precision, indicating its high accuracy in recognizing Bangla license plates.

In their work[59], Liu and al proposes a license plate recognition method that uses connected component analysis, projection analysis for segmentation, and two CNNs for character recognition. It can handle overexposed plates by converting them to grayscale and enhancing contrast ratio, and then using a breadth-first search algorithm to identify missing or redundant characters. Two CNNs were developed for character recognition, one

for Chinese characters and the other for numbers and letters. These simple and recurrent CNNs were trained on a dataset of 2189 images. The results showed a segmentation rate of 96.58% and recognition rate of 98.09%, outperforming previous methods.

2.5.2 Recurrent Neural Network (RNN)

In[60],Li and Shen have proposed a CNN-RNN model for the automatic license plate recognition (ALPR). Their method uses a CNN to extract visual features from the plaque picture, followed by an RNN with long-term memory units (LSTM) to interpret the character sequence. This combination has allowed them to achieve 96.5% accuracy on a data game that captures plaques in real-world environments. This technique has been especially effective in treating partially occult or floating pictures.

In[61], Cheang and al. have created a method based on the ConvNet-RNN model, in which an RNN is used to recognize the characters without the need for explicit segmentation, and a CNN is used to detect and extract the image's license plate. Compared to traditional approaches, their method has allowed for both a reduction in processing time and an improvement in recognition precision. This architecture has demonstrated its effectiveness on a variety of data sets and achieved a precision of over 97% under a range of situations, including partially visible plaques and obscured angles.

In[62],Wang and al. have proposed a real-time ALPR model that combines two CNNs for plaque detection and recognition. While the second extracts the characters, which are then analyzed by an RNN, the first CNN finds the plaque. With an accuracy of over 99% on the CCPD and AOLP data sets, their method has enabled a relative improvement of 50% in error rate compared to existing models. This model has shown exceptional performance in identifying plaques under challenging settings, such as changing lighting and moving plates.

2.5.3 Optical Character Recognition (OCR)

In the study referenced in [63],An automated number plate recognition (ANPR) system was introduced by Kashyap, which recognizes license plate characters using image processing techniques. In order to extract the required data, including the license plate number, the system was built to automatically take and process pictures of license plates. The system made use of optical character recognition (OCR) technology to identify the characters on the license plate. OCR is a procedure that enables the system to read and recognize the license plate number by converting the characters on an image to text.
Using a testing dataset, Kashyap assessed the effectiveness of the suggested ANPR system and found that it had an accuracy of roughly 82.6%. Even though this accuracy level is quite good, there is still opportunity for development because it could not be adequate for some applications, including toll collecting or law enforcement.

In [64], the study introduces two efficient Automatic License Plate Recognition (ALPR) methods for videos: Visual Rhythm (VR) and Accumulative Line Analysis (ALA). The VR method creates condensed time-spatial images to detect vehicle crossings, using YOLOv8 for license plate detection and a CNN-based OCR model for text extraction. The ALA algorithm tracks vehicles by analyzing a single video line via background sub-

traction and logical operations. Results show VR is more robust, achieving 96% F-score in challenging lighting and 88.7% OCR accuracy at 63 FPS, comparable to traditional methods. ALA is faster (74 FPS) but less accurate (80.8%). Both methods reduce computation by processing only one frame per vehicle, making them efficient for real-time ALPR.

In the study mentioned in [65],Rehman presented a novel approach to vehicle number plate recognition in Pakistani using template matching techniques and optical character recognition (OCR). The suggested approach was tested using many real-time photos of Pakistani license plates in various formats. The main goal of the suggested approach was to create an Automatic Number Plate Recognition (ANPR) system that would improve home security and save time and money for both private organizations and law enforcement. The suggested ANPR system recognized the characters on the license plate and turned them into machine-readable text using OCR technology in order to accomplish this goal. In order to precisely identify the license plate number, the system additionally employed template matching techniques to compare the identified characters with a predefined database of license plate characters. Rehman reported that their proposed ANPR approach achieved an accuracy of 93%, indicating that the system was able to identify the license plate numbers accurately in 93% of the case.

2.5.4 Template Matching

In[66],The characters split from Moroccan style number plates were recognized using the Template Matching approach. In order to correctly recognize the segmented characters, this approach compares them to a pre-established template of characters. Four distinct sets of Moroccan format number plates were used to test the suggested system; the corresponding success rates were 98.1%, 96.37%, 93.07%, and 92.52%. These outcomes demonstrate how well the Template Matching method works to identify characters on Moroccan type license plates. It is important to keep in mind, nevertheless, that this approach might not be as reliable as other deep learning-based techniques and might not be able to handle variations in license plate formats and lighting circumstances.

2.6 Vehicle Identification and Feature Recognition

In this section, we focus on the studies that develop methods for vehicle feature recognition, which includes model recognition, color detection, orientation and side detection, and vehicle type detection.

2.6.1 vehicle detection and type recognition

In [67], for vehicle detection and recognition, employing a multi-stage approach. First, video frames are converted into images (470×360 pixels) and preprocessed using median filtering (6×6 grid) and background subtraction. Object shape optimization is performed using change detection and Gaussian Mixture Model (GMM) to refine vehicle silhouettes. Two key features—energy features (power index-based matrix) and dense optical flow (motion analysis)—are extracted, followed by graph mining to optimize feature selection. Finally, an Artificial Neural Network (ANN) classifies vehicles. The system achieves a 93.75% mean recognition rate on the LISA dataset (95.62% TPR on LDB1, 91.89% on

LDB2) and 82.85% on KITTI, outperforming methods like ALVeRT and PointNet.

In [68], the authors proposed a vehicle type classification system using Principal Component Analysis (PCA) combined with self-clustering. The method begins by extracting the vehicle's front view using the license plate's location, followed by eigenvector generation to represent vehicle features. These features are then clustered via an adaptive self-clustering approach to distinguish between vehicle types. Tested on a dataset of 4,924 front-view vehicle images captured under various conditions, the method achieved a classification accuracy of 95.1%, outperforming traditional PCA and baseline classification methods.

2.6.2 vehicle color detection and classification

In [69], Rachmadi and Purnama , demonstrated that Convolutional Neural Networks (CNNs), typically used for shape recognition, can effectively perform vehicle color classification. Their approach processed images in multiple color spaces (RGB, HSV, CIE Lab, and CIE XYZ) through a dual-network CNN architecture with 16 layers. The model achieved 94.47% accuracy on a dataset of 15,601 vehicle images across 8 color classes, while maintaining real-time performance (0.156 sec/image on GPU). The results showed CNNs' strong capability in color-based classification, though some challenges remained in distinguishing similar colors under varying lighting conditions. This work highlights CNNs' potential for intelligent transportation applications where color recognition is crucial.

In [70] Chen and al, developed a vehicle color recognition system based on Feature Context (FC) for urban road applications. Their method first divides vehicle images into strategically configured subregions, then extracts color histograms from each region. These histograms are processed using a Bag-of-Words approach, where values are clustered to form a compact visual vocabulary. The system employs linear SVM for classification, combined with preprocessing techniques including haze removal and color contrast normalization to enhance robustness under varying environmental conditions. This FC-based approach achieved recognition accuracy exceeding 92%, demonstrating effectiveness for real-world traffic monitoring scenarios where computational efficiency and spatial awareness are crucial.

Baek and al, in [71] presented a vehicle color classification system utilizing 2D histogram features in HSV color space combined with Support Vector Machine (SVM) classification. Their approach specifically focused on the hue and saturation channels from the HSV color model, constructing a two-dimensional histogram that captures the joint distribution of these color attributes. The system processed 500 outdoor vehicle images across five distinct color classes: black, white, red, yellow, and blue. For classification, they employed an SVM classifier trained on these 2D histogram features, which effectively mapped the color distribution patterns to the respective color categories. The experimental results demonstrated high performance, achieving an average classification accuracy of 94.92%. This method proved particularly effective for outdoor vehicle color recognition while maintaining computational efficiency suitable for real-world applications.

2.6.3 vehicle side, orientation Detection & Classification

In this work[72], the author Youngmin and al present a two-step approach for vehicle orientation detection using synthetic data enhanced by photo-realistic image generation. The methodology involves constructing a meta-table containing class labels for vehicle orientation (front/back/side) from synthetic datasets, then transforming synthetic images into photo-realistic styles using CUT (Contrastive Unpaired Translation). The detection process uses YOLOv5 for initial localization followed by EfficientNet-b7 for fine-grained orientation classification across 3 orientations (front, back, side) using the enhanced meta-table combining original and translated images.

In this study [73], the author Rybski and al worked with a small dataset of 284 vehicle images taken from eight different angles (front, side, rear, and diagonal views). They used Histogram of Oriented Gradients (HOG) features and trained Support Vector Machine (SVM) classifiers for each specific orientation. To make the predictions more reliable, they applied Platt scaling to turn SVM scores into probabilities and picked the most likely orientation. The system achieved an 88% accuracy, although there were some mistakes. Interestingly, a single, unified classifier that considered all views together performed better than the separate ones it reached a higher ROC AUC score of 0.92 compared to 0.85, and it was also about 8× faster.

2.6.4 vehicle make (logo) detection & recognition

In [74], Kunduraci and Örnek (2019) proposed a Faster R-CNN (ResNet-50) system for vehicle brand detection, trained on 1,557 images of 20 common Turkish vehicle brands. The model achieved 67.66% accuracy on 1,280 test images, excelling for distinct logos like Fiat (100%) but struggling with similar-looking brands like Ford (1.22%) and Mercedes (5.45%). Performance improved to 78.58% when excluding these problematic cases. The results highlight the challenges of fine-grained brand recognition, particularly with visually similar logos, suggesting the need for enhanced preprocessing or larger training datasets to improve differentiation between comparable brands.

In [75], the authors propose an improved YOLOv4 model for vehicle logo detection, addressing challenges such as small object size, irregular shapes, and complex backgrounds. The backbone network was enhanced by partially replacing CSPDarknet53 with CSP-DenseNet, improving feature reuse and gradient flow. In the neck structure, a shallow output layer was added to retain small object features, and deformable convolutional blocks (res-dcn) were introduced to better handle irregular logo shapes. A new detection head (CT-Head) combines CNNs and Transformers to capture both local and global features, with residual connections to improve gradient stability. Additionally, an EA-Net attention mechanism helps focus on relevant logo features while suppressing background noise. Evaluated on the VLD-45 dataset (45 categories, 30,000 images), the model achieved a mAP of 62.94% (5.72% higher than original YOLOv4), a recall of 67.63%, and an APS of 58.9%, outperforming SSD, RetinaNet, YOLOv5, and others, especially for small object detection.

2.6.5 vehicle model recognition

In [76], the authors introduced a Coarse-to-Fine Convolutional Neural Network (CF-

CNN) for fine-grained vehicle model recognition. The approach starts by using a pre-trained CNN (similar to AlexNet) to extract global features and classify vehicles into broad categories. It then refines recognition by detecting discriminative local parts—like logos or headlights—through heatmaps from the CNN's last convolutional layer. These local features are mapped to the input image and further refined in a hierarchical manner. A one-vs-all linear SVM is trained on the combined global and local features. The model was fine-tuned using the CompCars dataset, containing 44,481 frontal-view images of 281 vehicle models. The proposed CF-CNN achieved a recognition accuracy of 98.29%, outperforming handcrafted-feature methods (like SIFT, HOG) and non-hierarchical CNNs. It also demonstrated strong generalization by achieving 90% accuracy on a secondary dataset with different viewpoints.

In [77], the authors presents the CompCars dataset, a large-scale collection of over 208,000 images of 1,716 car models, annotated with viewpoints, parts, and attributes. For vehicle model recognition, the authors fine-tuned CNNs (AlexNet, Overfeat, GoogLeNet) pre-trained on ImageNet. Their approach combined multi-view learning (using all angles), part-based recognition (e.g., headlights, taillights), and attribute prediction (e.g., door count). The best accuracy—89.6%—was achieved with GoogLeNet on the full dataset, while part-based voting improved top-1 accuracy to 80.8% on 431 models. For surveillance (frontal-view) data, they reached about 95% accuracy, and for model verification, they obtained 90.7% accuracy.

2.7 Conclusion

This chapter has outlined the key techniques for automated vehicle recognition, highlighting the shift from traditional image processing to deep learning methods like CNNs and YOLO. While earlier methods focused on license plate recognition, modern systems now identify additional vehicle features such as make, model, color, and orientation. Deep learning models offer superior performance in these tasks, especially under challenging conditions. While traditional techniques like OCR and template matching still have their place, deep learning approaches provide more accurate, scalable, and adaptable solutions. These advancements are critical for applications in traffic management, law enforcement, and smart cities, enabling efficient and reliable automated vehicle recognition.

Chapter 3

Conception

3.1 Introduction

In this chapter, we propose an integrated approach for vehicle recognition using deep learning techniques. Our system combines the power of the YOLO algorithm for the detection of vehicles, license plates, license plate characters, and vehicle orientation, along with CNN-based models for the recognition of vehicle color, brand (logo), and specific model. The goal is to efficiently detect vehicles and extract detailed information about them, including their license plate content, orientation, brand, color, and model. By leveraging deep learning, our system effectively handles various challenges such as different vehicle types, plate styles, lighting variations, and occlusions. This chapter presents the objectives of our system, its architecture, the implementation steps, and the proposed algorithms. We conclude with insights into the potential applications of our comprehensive vehicle recognition approach.

3.2 Problems and system goals

The essential problems in this study that make it difficult and challenging to achieve high accuracy in recognizing vehicles can be presented as follows:

1. The complexity of preprocessing and feature extraction from vehicle images and videos due to varying conditions and perspectives.
2. The high computational cost of training deep learning models on large and diverse datasets.
3. Difficulty in capturing subtle differences between similar vehicle models and brands.
4. The limitation of video-type input vehicle recognition systems, which may affect frame quality and temporal consistency.
5. The lack of research studies focused on Algerian vehicle recognition, which limits the scope for local adaptation and performance evaluation.
6. The limited availability of publicly accessible of Algerian vehicle and license plate datasets for training and evaluation.

7. Environmental challenges like motion blur, poor lighting, occlusion, and reflections that degrade image/video quality.

8. Variation in vehicle orientations, angles, and occlusion in real-world scenarios.

9. Privacy and data collection limitations that restrict dataset size and diversity.

Based on these problems, we proposed solutions to address the presented challenges. The primary goal of this project is to create a deep learning-based system capable of detecting vehicles in input images or videos and extracting comprehensive information about them. This includes detecting and localizing the vehicle, its license plate, plate characters, and orientation using YOLO, as well as recognizing the vehicle's color, brand (logo), and model using CNN models. By leveraging advanced deep learning techniques, the system aims to identify and extract all relevant details, providing a complete and accurate understanding of each detected vehicle. Figure 3.1 presents the general architecture of our system.

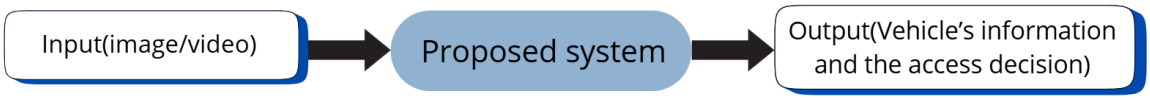


Figure 3.1: The general form of the proposed architecture of our system.

3.3 Characteristics of Algerian plates

The Algerian license plate contain ten or eleven numbers as showing in the image bellow:

Figure 3.2: Algerian license plate.

- The first two-digit identifies the wilaya or province in which the vehicle was first registered.
- The second tow-digit indicate the year in which the vehicle was manufactured.
- The third digit in the 3-digit group indicates the class of the vehicle.
- The rest of the number indicate the vehicle's serial number.

3.4 Overview of Utilized Datasets

To develop and evaluate our vehicle recognition system, we collected and curated a diverse set of datasets that covers the proposed system phases. These phases include vehicle

detection, license plate localization and recognition, logo classification, color and direction prediction, and vehicle model identification. Each part requires specific types of data, making it necessary to source and integrate multiple datasets from various domains, which reflects the complexity and multi-stage nature of our system; The datasets was collected for the train, validate, and evaluate models purposes. The following subsections describe these datasets in detail, along with their key characteristics, sources, and relevance to each stage of our system.

1. **Vehicle Dataset for YOLO Dataset (Vehicle Detection & Recognition Phase)**
   This dataset [78] contains 10,366 images from six different classes including car, motorbike, van, bus, truck, and bicycle. The dataset was divided into two distinct folders: a training folder (8,267 images) and a validation folder (2,099 images). This division ensured a well-balanced distribution of data for training and evaluation purposes. All annotations follow the YOLO format, with bounding boxes identifying object classes. Figure 3.3 presents a sample of the images that exist in this dataset.

Figure 3.3: Sample images from the Vehicle Dataset for YOLO

2. **AT HACKATHON (License Plate Detection Phase)**
   This selected dataset is available on Roboflow Universe [13]. It contains 1,846 annotated images of Algerian license plates captured under diverse weather, lighting conditions, and angles. The dataset is divided into a training folder (1,476 images) and a validation folder (370 images). Figure 3.4 presents a sample of the images that exist in this dataset.

Figure 3.4: Sample images from the AT HACKATHON dataset

3. **ALKO1 (Character Detection & Recognition Phase)**
   The dataset hosted on Roboflow under the name ”plate detector” (ALKO1) [79] includes 3,041 images with 36 classes representing digits (09) and uppercase letters (AZ). Each character is annotated using YOLO format with bounding boxes and class labels. The dataset is organized into training (2,422 images), validation (303 images), and test (316 images) sets. Configuration files (e.g., data.yaml) are provided to guide model training and class mapping. Figure 3.5 presents a sample of the images that exist in this dataset.

Figure 3.5: Sample images from the ALKO1 dataset

4. **Cars Logo Dataset (Logo Detection Phase)**
   The dataset used for logo detection is available on Roboflow [80]. It comprises 1,683 images with annotated bounding boxes for car logos across multiple brands (e.g., Toyota, BMW, Audi, Hyundai, Renault, Mercedes). It is organized into training (1,346 images), validation (168 images), and test (169 images) subsets. The dataset is annotated in YOLO format and includes logos in varied contexts such as urban roads and parking areas. Figure 3.6 presents a sample of the images that exist in this dataset.

Figure 3.6: Sample images from the Cars Logo dataset

5. **Car Logos Dataset (Logo Recognition Phase)**
   The logo recognition dataset from Kaggle [81] contains 8,252 preprocessed and labeled logo images. Images are divided into folders named after each brand (e.g., Audi, Honda, Ford), and further split into training, validation, and test sets. The dataset includes lighting variations, occlusions, and diverse logo presentations. Figure 3.7 presents a sample of the images that exist in this dataset.

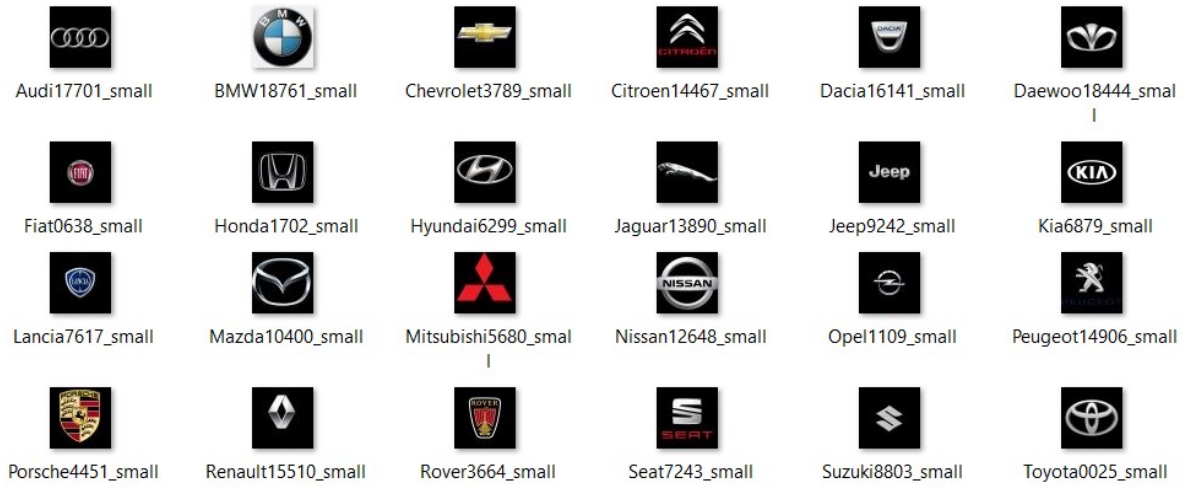


Figure 3.7: Sample images from the Car Logos dataset

Upon downloading the database, we discovered a number of defective photos. Some

photos were incorrectly categorized; for instance, an Opel was mistakenly identified as a Skoda. Additionally, we saw pictures of coffee cups with several designs or T-shirts with a logo on the left, as well as pictures with four logos that were repeated in every class. Figure 3.8presents a sample of misclassified and irrelevant cases that exist in this dataset.

Figure 3.8: Example of misclassified and irrelevant images in the Car Logos dataset

Therefore, in order to get better outcomes, we started by deleting these photographs. Once the images containing false information were removed, we plotted the rectangles for each class to visualize the number of images available. We noticed an imbalance in our database, which led us to balance it using techniques such as augmentation for small classes by applying transformations like (rotation, flipping, scaling), and downsampling, which randomly removes samples from overrepresented classes to match the size of minority classes. Figure 3.9 shows the effect of balancing the dataset.

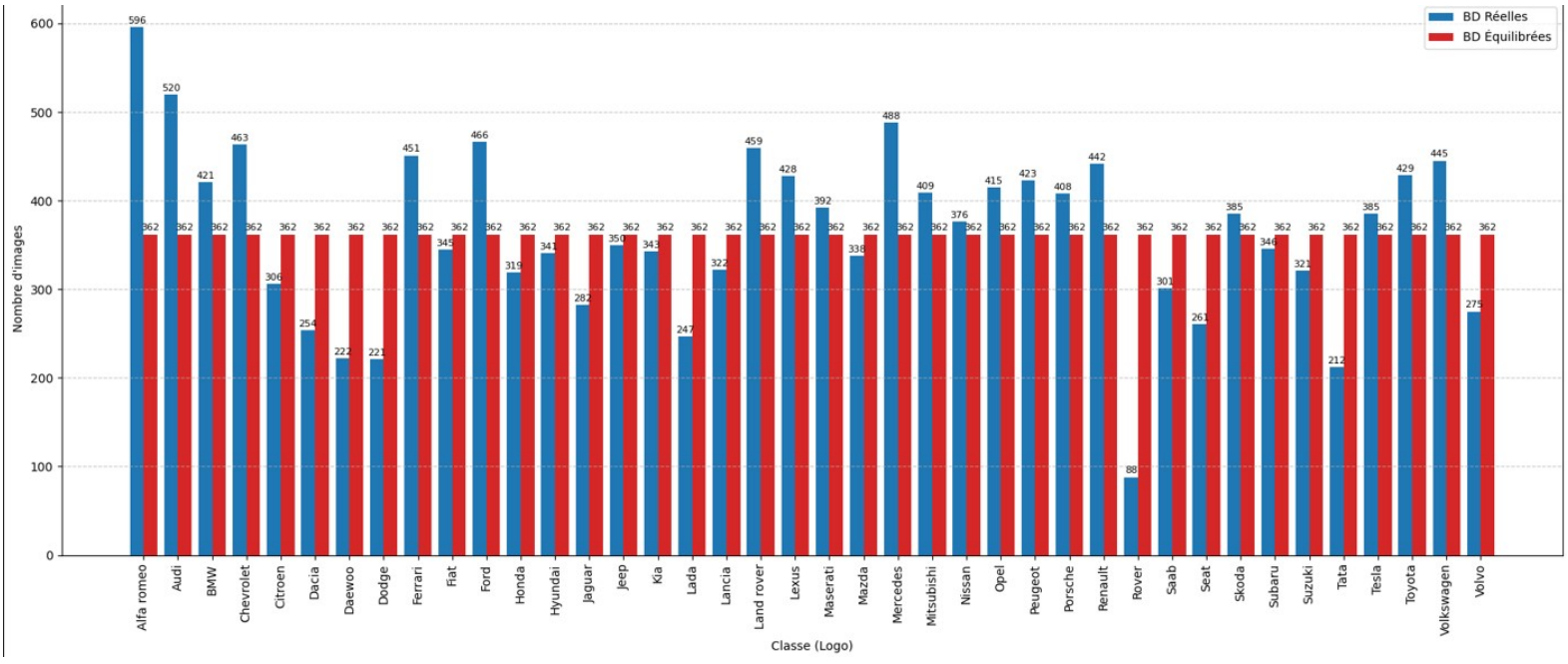


Figure 3.9: Visualization of class distribution before and after balancing

6. **Car Make, Model, and Generation Dataset (Vehicle Model Recognition Phase)**
   We obtained this dataset from Kaggle [82], containing over 41,521 labeled images. The dataset includes metadata for make, model, and generation. For our task, we extracted and retained only the model labels (e.g., Toyota-Models/, Chevrolet-Models/, Nissan-Models/...), discarding generation data. We then applied augmentation to underrepresented brand folders to expand and balance the dataset, ensuring robust training coverage. Figure 3.10 presents a sample of the images that exist in this dataset.

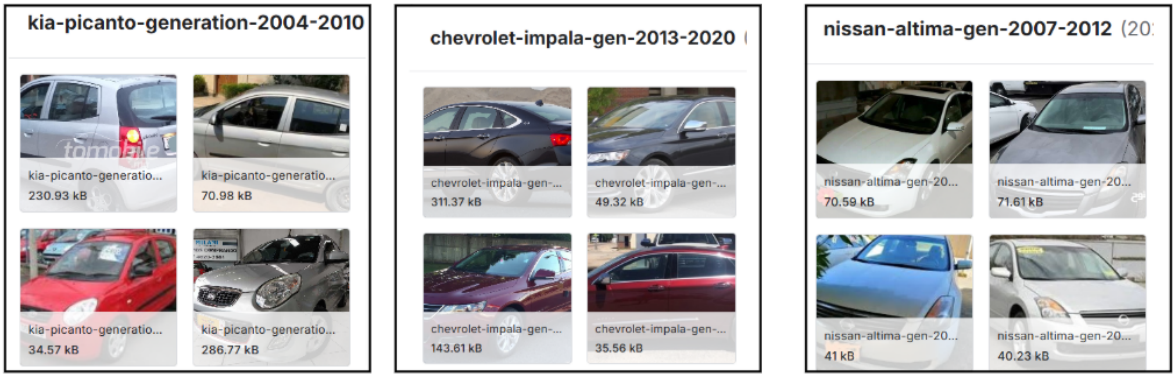


Figure 3.10: Sample images from the Car Make, Model, and Generation dataset

7. **VCOR (Vehicle Color Classification Phase)**
   We used the VCOR (Vehicle Color Recognition) dataset from Kaggle [83], which includes 11,382 images categorized into 15 predefined vehicle color classes. The dataset is annotated with color labels such as white, black, red, blue, and silver. This labeled dataset provides the basis for training a vehicle color classifier. Figure 3.11 presents a sample of the images that exist in this dataset

Figure 3.11: Sample images from VCOR dataset

8. **Front and Rear of Car (Direction Prediction Phase)**
   We utilized the "Front and Rear of Car" dataset from Roboflow [84], containing

1,082 annotated images labeled as either ”front” or ”rear”. Each image includes a bounding box and class label. The dataset’s consistent annotations and image variety (in angles and lighting) support training an orientation classifier. Figure 3.12 presents a sample of the images that exist in this dataset.

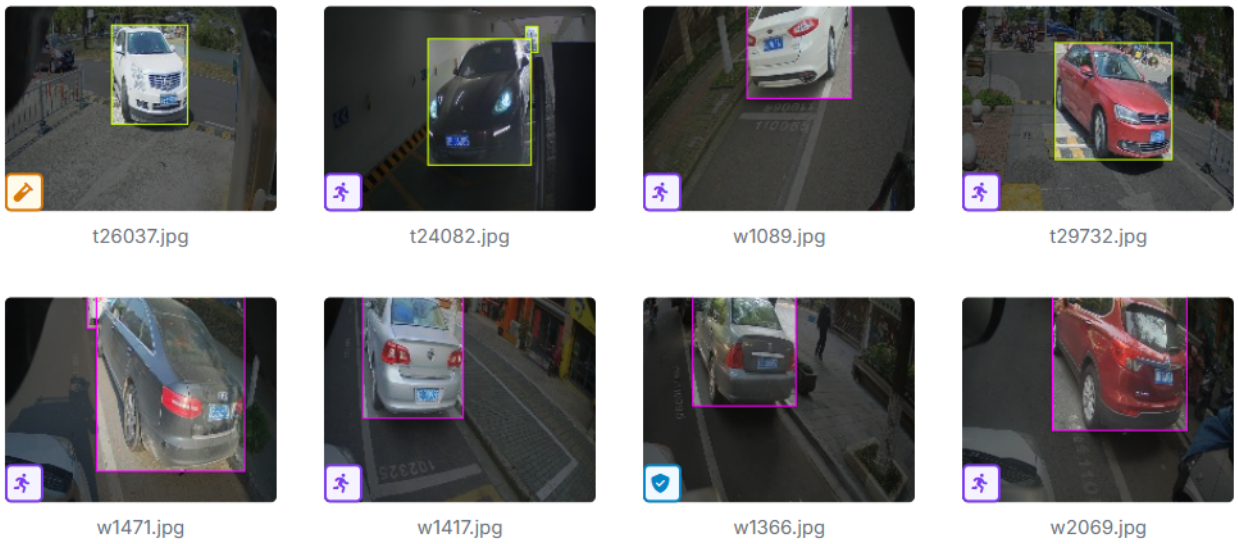


Figure 3.12: Sample images from the Front and Rear of Car dataset

3.5 Architecture of the system

The architecture diagram below illustrates how this comprehensive workflow enables accurate detection of vehicles and extraction of detailed information, including license plate localization, character recognition, vehicle orientation, as well as classification of the vehicle’s color, brand (logo), and model from image and video inputs.
The proposed system consists of the following phases:

1. **Vehicle Identification**

   - Vehicle Detection: This phase detects the presence of a vehicle in the input image or video using an object detection model. It accurately locates the vehicle’s boundaries, even in complex backgrounds or partial occlusions.
   - License Plate Detection: This phase identifies the license plate within the detected vehicle region, even under challenging conditions such as skewed angles, poor lighting, or partial occlusions.
   - License Plate Preprocessing: In this phase, filtering, contrast enhancement, noise reduction, and geometric correction techniques are applied to the detected license plate image to prepare it for further processing.
   - Character Detection & Recognition: The preprocessed license plate image is used to segment individual characters. These characters are then digitized using an optical character recognition (OCR) model.
   - Plate Classification: Each recognized plate number is classified into a algerian or not.

- Color prediction: This phase uses a classification model to predict the color of the vehicle from one of the predefined 15 color classes.
- Direction prediction (front/rear): A binary classification model determines whether the vehicle is facing the front or rear direction based on its visual features.
- Logo Detection: The car manufacturer's logo is detected within the image using a dedicated object detection model, focusing on the region typically located on the front or rear of the vehicle.
- Logo Extraction and Preprocessing: The detected logo is extracted and preprocessed (resizing, normalization, etc.) to prepare it for classification.
- Logo Recognition (Logo Classification): The preprocessed logo image is passed through a CNN-based model to classify it into one of the known car brand logos.
- Recognized car model: Based on the detected logo and additional visual features, the specific model of the vehicle is predicted using a trained model classifier.
- Vehicle data: This includes all extracted and recognized information about the vehicle: license plate number, brand logo, color, direction, and model.

2. **control Access**

- Existing Data: This refers to previously stored records in the system database, including authorized vehicles, their owners, and access history.
- New Data: This represents newly captured and processed vehicle data from the current input, not yet recorded in the database.
- Access Verification: The system compares the new data with the existing database to verify if the vehicle is authorized to access the area.
- Authorization case selection: Based on the verification result, the system selects the appropriate access decision—grant, deny, or request manual review.
- Saving data: All processed data (recognized features, access decisions, timestamps) are saved to the system's database for logging and future reference.
- Action decision: The final decision is taken based on access verification and classification results.

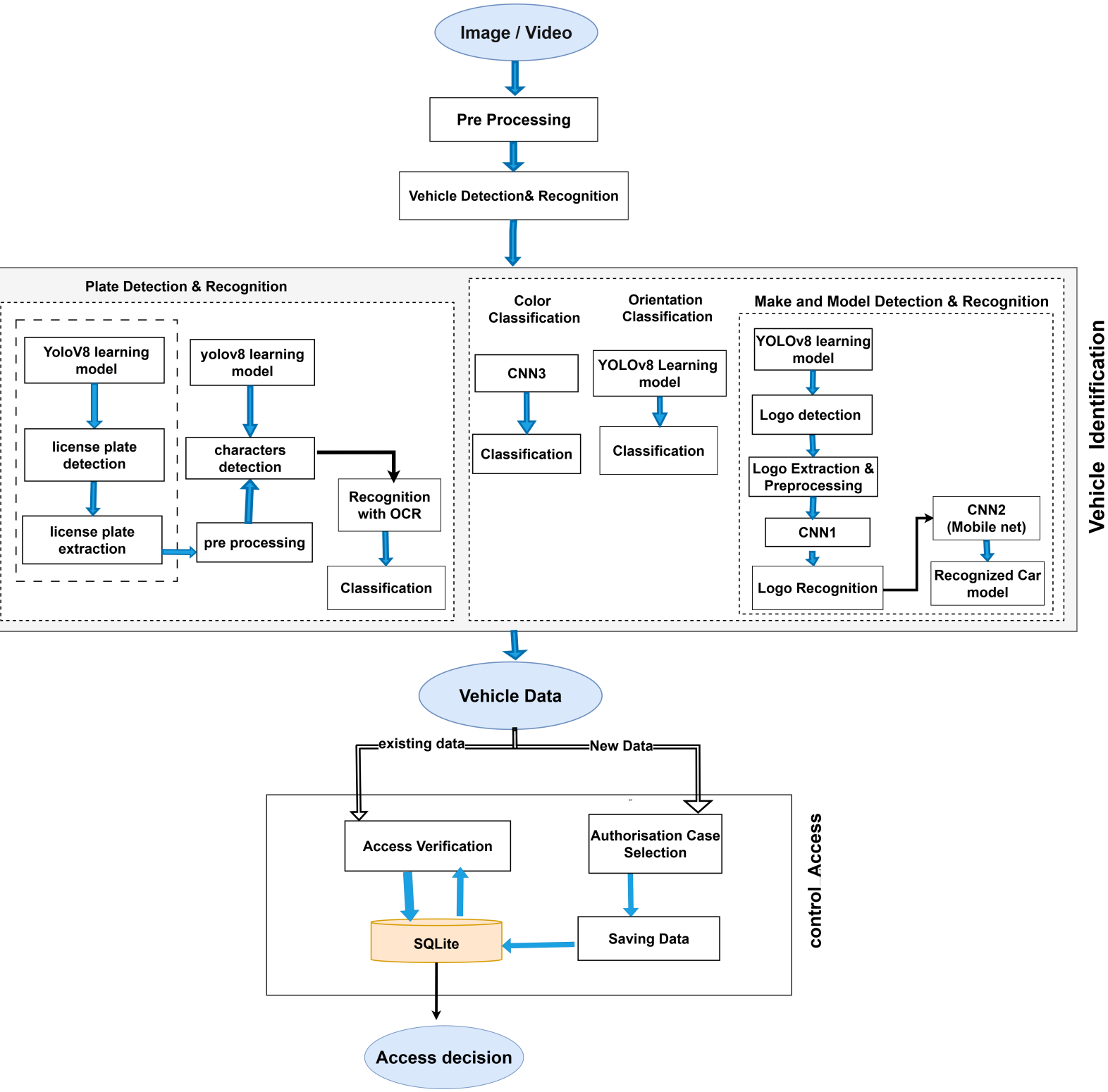


Figure 3.13: The architecture of the proposed vehicle recognition system.

3.6 Preprocessing phase

The first phase in our system is preprocessing the provided input, which can be either a video or an image. In this phase, we encountered difficulties in obtaining videos that align with the concept of our system, as well as suitable images to ensure accurate results in the subsequent stages. Hence, we managed to apply various preprocessing techniques for image/video, which enhance image clarity, reduce noise, and improve the input's suitability for tasks (e.g., license plate or vehicle model recognition) and character recognition. Below is a step-by-step explanation of each operation:

3.6.1 Frame extraction

Considering that the videos used in our system have duration start from [0.03 to 0.05] second, we conducted numerous tests to determine the appropriate number of frames that can be extracted from each video; we have concluded that 10 frames are the most appropriate for our system, with each frame representing the complete vehicles.In Figure 3.14, we demonstrate dividing one video into 10 frames and converting each frame to RGB format.

Figure 3.14: Example of dividing video into frames.

1. **Color Normalization**
   The objective of this step is to enhance the visual quality of the image by increasing its local contrast, especially in cases of low-light or uneven illumination[85].
   The process begins by converting the input image I BGR from the BGR color space to the perceptually uniform LAB space, as expressed in Equation 3.1:

$$\text{LAB} = T_{\text{BGR}\to\text{LAB}}(I_{\text{BGR}}) \quad [[86]] \tag{3.1}$$

   In this color space, the L channel (lightness) is the focus of contrast enhancement. We apply Contrast Limited Adaptive Histogram Equalization (CLAHE) to the lightness component to improve local contrast while suppressing noise amplification. The enhancement is defined as:

$$L' = \text{CLAHE}(L;\Theta), \quad \Theta = \{\text{clipLimit} = 2.0, \text{tileSize} = 8\times 8\} \quad [[87]] \tag{3.2}$$

This method enhances the contrast in localized regions of the image, with the clipLimit controlling the maximum contrast to avoid over-enhancement, and tileSize defining the size of the regions for adaptive equalization. Once the lightness channel has been enhanced, it is merged back with the original A and B channels to reconstruct the color image in BGR space, preserving the original color fidelity. This is described by

$$I_{\text{normalized}} = T_{\text{LAB}\to\text{BGR}}(\{L', A, B\}) \tag{3.3}$$

Figure 3.15represents the original image before processing, while Figure 3.16 illustrates the result after applying the color normalization procedure.

Figure 3.15: Original Picture

Figure 3.16: After Color Normalization

2. **Noise Reduction**
   In the second step, noise reduction is performed using the Non-Local Means Denoising algorithm. The main goal here is to clean up the image by reducing noise, especially in areas with solid colors or textures, without blurring important details like edges or text. This method works by comparing small patches across the image and averaging similar ones, which helps to remove unwanted variations while keeping sharp features intact.
   For an image I, the denoised value at pixel i is computed as:

$$I^{\text{denoised}}(i) = \frac{1}{Z(i)} \sum_{j \in \Omega} w(i,j) \cdot I(j) \tag{3.4}$$

   where:

   - $w(i,j) = \exp\left(-\frac{\|\mathcal{P}(i)-\mathcal{P}(j)\|^2}{h^2}\right)$ is the Gaussian weight measuring similarity between patches $\mathcal{P}(i)$ and $\mathcal{P}(j)$
   - $h = 10$ controls the decay of weights (for both luminance and color channels)
   - $Z(i) = \sum_j w(i,j)$ is the normalization factor

- $\mathcal{P}(i)$ denotes a 7×7 patch centered at pixel i
- Ω represents a 21×21 search window

As a result, the image becomes clearer and easier to process in later stages, such as vehicle detection, license plate detection and character recognition.

Figure 3.17represents the image before processing, while Figure 3.18 illustrates the result after applying the noise reduction procedure.

Figure 3.17: Before noise reduction

Figure 3.18: After noise reduction

3. **Edge Sharpening**
   The third step in our preprocessing pipeline is edge sharpening , which aims to enhance the boundaries of key objects, which in our case are the vehicle, its license plate, and the characters. This step improves the clarity of these features, helping the system better detect and recognize them in later phases. Given an input image I, the sharpened output Ix is obtained by applying a 2D convolution with the following kernel:

$$I^{\sharp} = I * K \tag{3.5}$$

where K is the sharpening kernel: [88]

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

This operation increases the distinction of the edges, making the relevant features more visible and easier for the recognition algorithms to process. (see Figure 3.20)

Figure 3.19: Before Edge Sharpening

Figure 3.20: Afer Edge Sharpening

3.7 Vehicle Detection & Recognition

The second phase of our system involves detecting and recognizing the vehicle present in the preprocessed image or video. For this task, we selected the YOLOv8 model[89]due to its efficiency and performance in real-time object detection. This model is composed of three core components: Backbone, Neck, and Head.

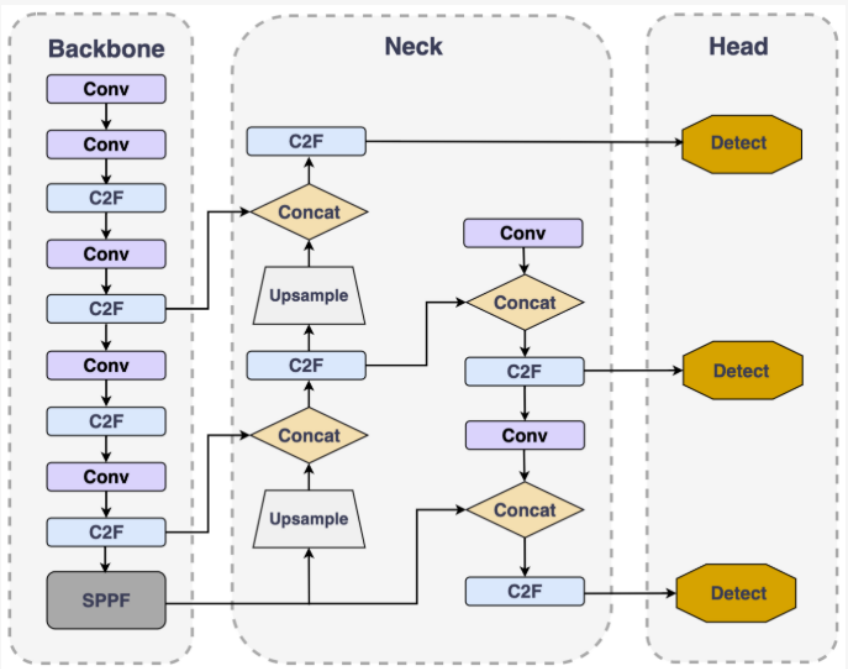


Figure 3.21: Key Component of YOLOV8.[90]

As illustrated in Figure 3.21, the model consists of the following parts:

Backbone: This component employs convolutional neural networks to extract key visual features and spatial patterns from the input image. It serves as the ”eyes” of the model, laying the foundational representation required for further processing.

Neck: This stage enhances the features obtained from the backbone by combining different feature maps using structures such as the Path Aggregation Network (PANet) and Feature Pyramid Network (FPN). These allow the model to better detect objects at multiple scales and improve contextual understanding.

Head: This is the final stage where object predictions are made, including bounding box coordinates and class probabilities. It plays a critical role in determining the model's speed and accuracy, as it directly influences what objects are detected and how reliably they are classified.[91], as shown in figure 3.22.

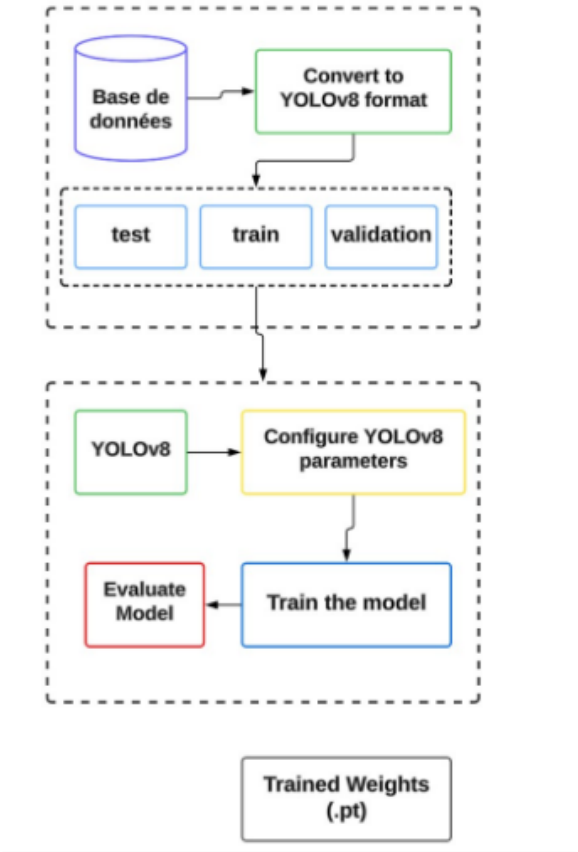


Figure 3.22: The YOLOv8 architecture.[92]

The data.yaml file is a configuration file used for training an object detection model with YOLOv8. It specifies the paths to the training and validation image directories and defines the six classes to be detected, which are: car, motorbike, threewheel, van, bus, and truck. These classes correspond to the Vehicle Dataset for YOLO [78](see section 3.4). The file provides the essential structure required to set up the training pipeline, including the number of classes and their corresponding labels.

The input images were resized to a fixed resolution of 640x640 pixels, enabling the model to process uniform input sizes and optimize detection across varying vehicle dimensions. The batch size was set to 16, which means that the model processes 16 images per training step. This setup helps to efficiently utilize hardware resources and stabilizes gradient updates.

Training was conducted over 70 epochs, where each epoch represents a complete pass through the entire training set. This iterative process allows the model to progressively learn feature representations and improve detection accuracy across all specified classes. The data.yaml file played a critical role in guiding the training process by mapping the image paths and class definitions.

After the training stage, the model was tested on a dedicated evaluation set to measure its detection performance. This testing phase allowed us to validate the model's generalization capabilities and ensure its robustness when applied to unseen images. Figure 3.23 presents the vehicle detection and recognition phase results sample.

Figure 3.23: The output of running model on a test image.

3.8 Vehicle Identification

The third phase covers vehicle identification, which includes Plate Detection and Recognition, Color Recognition, Orientation Detection, and Make and Model Recognition.

3.8.1 Plate Detection & Recognition

Here, we extract the vehicle's license plate and then recognize its content.

A. License plate detection

In this part, we use also YOLOv8 model for plate extraction, where we have just one class to be detected, which is "plaque". The file provides the necessary information for setting up the training process, including the class name and the number of class.

1. **Train model**
   The YOLOv8 model was trained for license plate detection using the AT Hackathon dataset (see Section 3.4). Input images were resized to 640×640 pixels, and the batch size was set to 16. Although training was configured for 50 epochs, it was early stopped at epoch 26 based on performance observations to prevent overfitting. Training progress was monitored using the results.csv file.

2. **Model Evaluation**
   During training, the model automatically saved the results of each epoch in two files: best.pt and last.pt.
   The best.pt file contains the model weights that gave the best performance on the validation set. To evaluate the model's performance, We used the best.pt file, since it represents the most accurate version of the model during training. This allowed me to verify that the model could provide reliable and accurate results on new, unseen data.

In this step, the YOLOv8 model is employed to detect license plates from input images. The process involves:

- **Model Initialization:** The trained YOLOv8 weights are loaded using the YOLO class from the Ultralytics library.
- **Inference:** The input image is passed to the model, which outputs bounding boxes, class labels, and confidence scores stored in the results object. Figure 3.24 presents the result of license plate detection using the YOLOv8 model.

Figure 3.24: An example of applying the selected model for license plate detection.

B. License plate extraction

The license plate region is extracted by utilizing the image with detection, which provides the necessary bounding box coordinates. The YOLOv8 model identifies the region of interest based on these coordinates. Subsequently, the extracted license plate is resized to the desired dimensions. This resizing operation ensures that the license plate image conforms to the specific size requirements for further processing. Finally, the resized license plate image is saved, ensuring its availability for subsequent steps in the workflow. Figure 3.25 presents the result of license plate extraction using the YOLOv8 model.

Figure 3.25: License Plate Extraction Result Using YOLOv8

C. License Plate Recognition

1. **Preprocessing of extracted image**
   These steps collectively preprocess the license plate image, enhancing its clarity, removing noise, and potentially correcting any misalignment caused by the plate's orientation.Here's a step-by-step explanation of each operation:

   (a) **Resize**
   The input image is resized to a specific size of 333 pixels width and 75 pixels height.
   This resizing step ensures that the license plate image has a consistent size for further processing. By enforcing a standardized size, the subsequent algorithms and models can operate on images with consistent dimensions, facilitating reliable and efficient processing.

   (b) **Grayscale**
   The resized image is converted from the BGR color space to grayscale . This conversion simplifies subsequent image processing steps.Figure 3.26 displays an example of a gray-scale license plate.

Figure 3.26: Example Of Gray-Scale License Plate

.

(c) **Histogram Equalization for Contrast Enhancement** Histogram equalization is a widely used image enhancement technique that improves the global contrast of an image. It works by redistributing the intensity values of pixels to span the entire available range. This is particularly effective in cases where the image's background and foreground are both bright or both dark.[93]

The transformation function used in histogram equalization is defined as:

$$s_k = (L-1)\sum_{j=0}^{k}\frac{n_j}{N} \tag{3.6}$$

where:

- s_k is the new intensity value after equalization.
- L is the number of possible intensity levels (e.g., 256 for 8-bit images).
- n_j is the number of pixels with intensity j.
- N is the total number of pixels in the image.

Figure 3.27: Example of Contrast Enhanced License Plate.

2. **Characters detection**
   For character detection, we employed a YOLOv8 model trained on the ALKO1 dataset (see Section 3.4), where each character (0–9, A–Z) is annotated with precise bounding boxes. The model localizes characters within license plate images by predicting bounding boxes for each symbol. Detections are sorted by horizontal position to reconstruct the correct sequence, and each character is extracted as an individual image for input to the recognition model.

   - **Train model**
     The input images are resized to a resolution of 640×640 pixels, allowing the YOLOv8 model to process images uniformly and effectively. A batch size of 16 was used, meaning that during each iteration of training, the model processes 16 images simultaneously, balancing computational efficiency and convergence speed.
     The training was configured to run for 30 epochs, with model weights automatically saved after each epoch using the save-period=1 parameter. This setup ensures that intermediate models are preserved, which is useful for analyzing training progress or resuming from a specific point if necessary. The training process was performed on a CUDA-enabled GPU (device='cuda') for faster computation, and utilized 5 parallel data-loading workers (workers=5) to speed up image loading and pre-processing.
   - **Model Evaluation**
     After training, the YOLOv8 model was evaluated using its built-in validation function, which measured key metrics such as precision, recall, and mean Average Precision (mAP). This evaluation on the validation set confirmed the model's ability to accurately detect and localize characters, ensuring it was ready for the recognition stage.

     An example of character detection results is presented in Figure 3.28

Figure 3.28: Example of characters Detection

3. **Characters Recognition**
   After detecting characters, each one is spatially segmented from the input image, producing a sequence of sub-images $\{I_1, I_2, \ldots, I_n\}$, where each Ii represents a single character candidate. These sub-images are then passed to a recognition model [94]. For this task, we use the TrOCR model that trained on the ALKO1 dataset (see Section 3.4), which presented as a function:

$$f : I_i \rightarrow c_i$$

   where f maps each character image Ii to its corresponding textual symbol $c_i \in A$, with A being the alphabet of possible characters (digits and capital letters). The complete license plate string P is then reconstructed by concatenating the recognized characters in order:

$$P = c_1 \parallel c_2 \parallel \cdots \parallel c_n$$

   This process eliminates the need for manual feature engineering or rule-based segmentation, this enables direct inference of the license plate string from the sequence $\{I_i\}$. The final result P serves as input for the subsequent classification stages. Meaning that once all characters in the list have been recognized, they are concatenated in the correct order to reconstruct the full license plate number, making the data ready for the classification (see Figure 3.29).

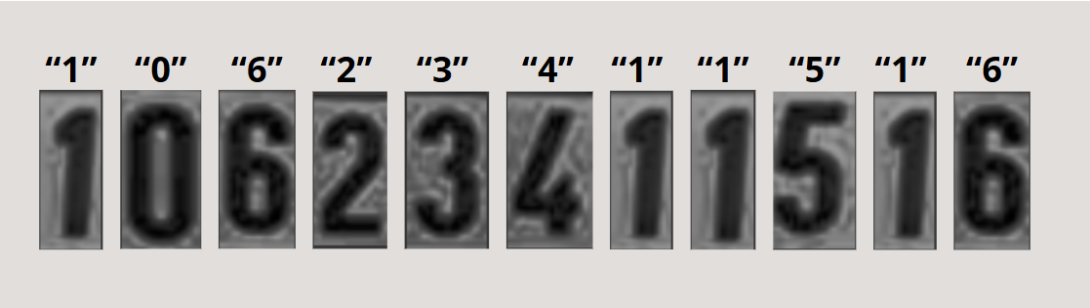
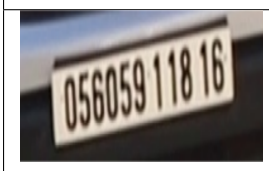


Figure 3.29: Result of Characters Recognition with TrOCR

Table 3.1 presents sample cases of both correct and incorrect detections, which subsequently led to correct or incorrect recognitions.

Table 3.1: License Plate Recognition Results

License Plate	Character Detection	Character recognition
		1562480016
		05605911816
		0597110443
		00 48
		0248101891 6

D. Classification

In this stage, we work with the extracted license plate number obtained after character recognition. Let the recognized license plate sequence be denoted as

$$P = c_1 \parallel c_2 \parallel \cdots \parallel c_n$$

with $c_i \in \mathcal{A}$ where $\mathcal{A}$ is the set of valid alphanumeric characters. The classification function is defined as:

$$\text{Classify}(P) = \begin{cases} f_{\text{ALG}}(P), & \text{if } P \in \mathcal{L}_{\text{ALG}} \\ \text{“Not Algerian”}, & \text{otherwise} \end{cases}$$

where $\mathcal{L}_{\text{ALG}}$ is the formal language representing valid Algerian license plate formats (e.g., patterns such as `NNNNNN-NN-NN` for standard plates), and $f_{\text{ALG}}(P)$ is a structured function that extracts and classifies the components of a given plate P (see Figure 3.30).

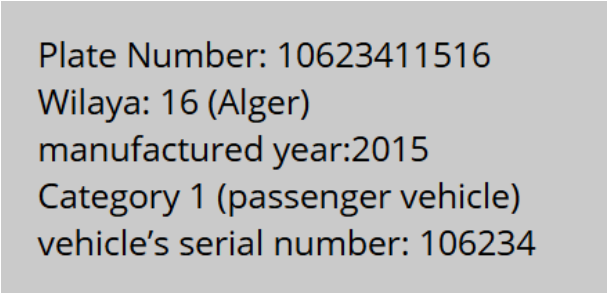


Figure 3.30: Example of PLate Classification

such as:

$w = f_w(P)$ → Wilaya code
$v = f_v(P)$ → Vehicle type
$y = f_y(P)$ → Registration year

This rule-based mechanism ensures that only license plate numbers conforming to Algerian structural norms are processed further. Any deviation from $\mathcal{L}_{\text{ALG}}$ results in a classification output of "Not Algerian". Figure 3.31 illustrates two examples: one of an Algerian license plate and one of a non-Algerian plate.

Figure 3.31: Examples of Algerian and Non-Algerian License Plates

3.8.2 Make and Model Detection & Recognition

1. **Logo Detection**
   At this stage, we also used the YOLOv8 model, trained on the Cars Logo dataset (see Section 3.4) using standard configurations suitable for object detection. The training was conducted over 80 epochs with a batch size of 16 and an initial learning rate of 0.001. To improve the model's generalization ability, data augmentation techniques such as horizontal flipping, brightness adjustment, and cropping were applied. This allowed the model to accurately detect logo locations regardless of image quality, orientation, or visual noise. Figure 3.32 presents the vehicle logo detection results.

Figure 3.32: Sample Results of Vehicle Logo Detection using YOLOv8

2. **Logo Extraction and Preprocessing**

   Following the YOLOv8 model's detection of the vehicle logos, the identified regions are taken out of the original image as Regions of Interest (ROIs). After being resized to a uniform 128 x 128 pixel size, each ROI is normalized to a [0,1] range and converted to grayscale. By minimizing noise and superfluous visual variation, this preprocessing guarantees consistent input size for the recognition model and highlights pertinent features like contours and brand shapes. This stage is essential for enhancing the logo recognition pipeline's robustness and performance. (see Figure 3.33)

Figure 3.33: Crop logo image.

3. **Logo Recognition**

   The proposed CNN architecture used for vehicle logo recognition is illustrated in Figure 3.34. The model was trained on the Car Logos Dataset (see Section 3.4), which includes 8,252 labeled images across various car brands, and is structured for classification tasks.

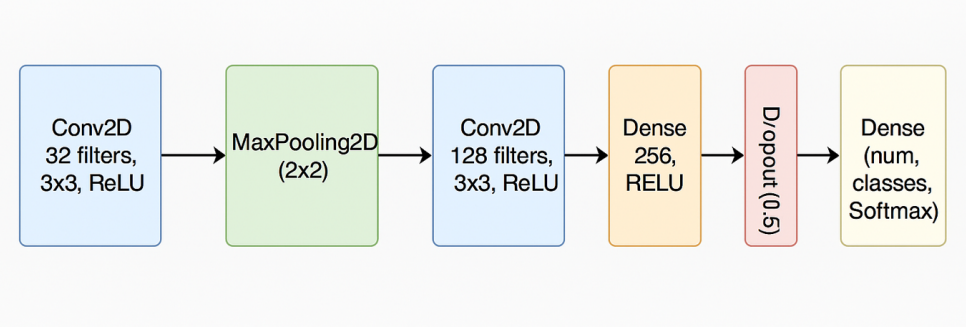


Figure 3.34: The proposed architecture of the CNN model for logo recognition

It begins with an input layer that receives RGB images resized to 128x128 pixels. The first convolutional layer applies 32 filters to extract basic features like edges and textures, followed by a max pooling layer that reduces spatial dimensions and prevents overfitting.

A second convolutional layer with 64 filters captures more complex patterns, again followed by max pooling.

A third convolutional layer with 128 filters extracts high-level features such as distinct logo shapes, and is also followed by max pooling.

The output is then flattened into a 1D vector and passed to a fully connected dense layer with 256 neurons that learns to combine the features for classification.

A dropout layer is used to randomly disable 50% of neurons during training, reducing the risk of overfitting.

Finally, the output layer uses a softmax activation to predict the probability of each logo class, allowing the model to classify the input logo accurately.

training the model:
The CNN model was trained on images resized to 128x128 pixels for optimal performance. Data augmentation techniques (rotation, zoom, horizontal flip) were applied to improve generalization. Training ran for 70 epochs with a batch size of 32. The Adam optimizer and categorical-crossentropy loss function were used, while a Dropout layer (0.5) helped prevent overfitting.

model evaluation:
Performance was monitored using a validation set during training. The model was then evaluated on a separate test set to objectively assess its accuracy and generalization capability. Key metrics including test accuracy and the confusion matrix were analyzed to measure real-world performance. Figure 3.35 presents a sample result from the brand recognition phase.

Figure 3.35: sample result from the brand recognition using CNN.

4. **Car Model Recognition(Two stage Approach)**
   For car model recognition, we adopt a smart two-step approach. First, the system detects the car brand logo (e.g., Toyota, Kia) using a logo recognition model trained on the Car Logos Dataset (see Section 3.4). Once the brand is identified, a specialized brand-specific classifier is activated to recognize the exact model of the vehicle. This model recognition phase is based on the Car Make, Model, and Generation Dataset (see Section 3.4). For example, if the detected logo is Toyota, the system invokes a classifier trained only on Toyota models (such as Corolla, Camry, Yaris, etc.). Similarly, if a Kia logo is detected, a dedicated Kia model recognizer is triggered. In the following a pseudocode that describes the car model recognition stage:

Algorithm 1 RecognizeCarModel(I)

Input: Image I
Output: Predicted car model
// Step 1: Detect the vehicle logo
$logo\_bbox, brand\_label \leftarrow LogoModel.predict(I)$
// Step 2: Crop logo region from image
$logo\_region \leftarrow Crop(I, logo\_bbox)$
// Step 3: Select brand-specific classifier
if $brand\_label \in BrandClassifiers$ **then**
  $model\_classifier \leftarrow BrandClassifiers[brand\_label]$
else
  return "Unknown Brand"
end if
// Step 4: Predict car model using selected classifier
$PredictedModel \leftarrow model\_classifier.predict(I)$
return $PredictedModel$

This approach improves accuracy by giving each brand its own expert system, makes training easier by separating different brands' models, and allows simple expansion since adding new brands just requires training new specialists without changing the existing system mirroring how humans naturally recognize cars by first identifying the manufacturer before narrowing down to specific models. (See Figure 3.37)

4-1. Isolated Model Training

We train dedicated classifiers for each brand with using a deep learning model based on MobileNetV2, a lightweight convolutional neural network architecture pretrained on ImageNet. The model is implemented using TensorFlow's Keras module [95]. The architecture includes the MobileNetV2 base (with top layers excluded), followed by global average pooling, fully connected layers, and dropout for regularization. This design allows the model to extract high-level features efficiently from input images and achieve accurate brand classification.
The proposed mobileNetV2 architecture that we used for vehicle model recognition is shown in the following figure.

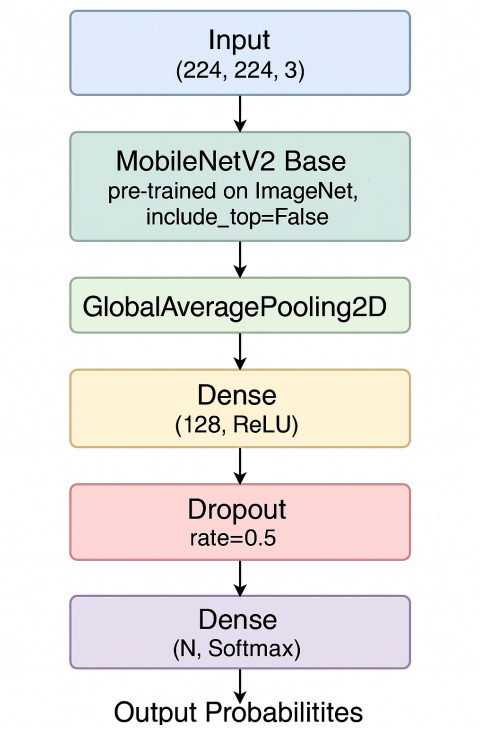


Figure 3.36: The proposed architecture of the mobileNetV2 model

In the following, the details of the provided architecture:

- **Input Layer**
  The input to the model is a color image of dimensions 224×224×3, corresponding to a resized RGB image. This input size is standard for pre-trained architectures like MobileNetV2, ensuring compatibility with pre-learned weights.
- **MobileNetV2 Base (Pre-trained)**
  The core of the architecture is based on MobileNetV2, a lightweight convolutional neural network optimized for resource-constrained devices. In our model, we use MobileNetV2 pre-trained on ImageNet with the parameter include-top=False, meaning the original classification layers are removed. This base serves as a feature extractor, capable of capturing rich visual patterns from the input images.
- **GlobalAveragePooling2D**
  Following the MobileNetV2 base, a GlobalAveragePooling2D layer is applied. This layer replaces traditional dense layers by averaging each feature map into a single value. It reduces the dimensionality of the data while retaining the most important information, helping to limit overfitting and improving the model's generalization.
- **Dense Layer**
  A fully connected Dense layer with 128 units and a ReLU activation function is added. This layer learns non-linear relationships from the features extracted by MobileNetV2. It plays a crucial role in transforming feature vectors into representations suitable for classification.
- **Dropout**
  To prevent overfitting, a Dropout layer with a rate of 0.5 is included. It

randomly disables 50% of the neurons during training, forcing the network not to rely too heavily on specific nodes and encouraging better generalization across unseen data.

- **Output Dense Layer (N classes, Softmax)** The final layer is a Dense layer with N units, where N represents the number of target classes in our classification task (e.g., different car brands or models). It uses a Softmax activation function to produce a probability distribution over the classes, allowing the model to predict the most likely class for each input image.

In the vehicle model recognition phase, we chose four specific car models because the dataset had enough images. Working with a small number of models also helped us keep the training more balanced and effective. We couldn't include other models since there weren't enough images of them in the dataset. Table 3.2 presents the available models for each brand.

Table 3.2: Common Vehicle Models per Brand in Algeria

Brand	**Models**
Chevrolet	aveo, impala, Malibu, Silverado, Tahoe, traverse
Nissan	Altima, armada, Datsun, maxima, navara, patrol, sunny
Kia	cadenza, cerato, optima, Sportage, Picanto, Rio
Toyota	Avalon, Corolla,Fj, Fortuner, hiace, hilux, innova, landcruiser, prada, rav4, yaris

(a) **KIA Model Recognition**
Tain the model: The model specifically trained on a dataset containing 6 distinct models (Sportage, Rio, Picanto, Cadenza, Cerato, Optima). The input images were resized to 224x224 pixels, and we used a batch size of 32 during training. At first, we kept the base model frozen and trained the new classification layers for 50 epochs using the Adam optimizer with a learning rate of 0.0001. After this initial training phase, we fine-tuned the model by unfreezing the top 30 layers of MobileNetV2 and trained it again for 10 more epochs with a lower learning rate (0.00001).

Model Evaluation: After training, we evaluated the model on a separate test set to see how well it could classify new, unseen images. The model achieved a high accuracy, indicating that it had learned to recognize the car models effectively.

(b) **Chevrolet Model Recognition Train the model :**
The model was trained using transfer learning with MobileNetV2 as the base architecture, processing images resized to 224x224 pixels. A batch size of 32 was used, with data augmentation applied to the training set (20% reserved for validation).
The training occurred in two phases: initial training with frozen layers (50 epochs) followed by fine-tuning (10 epochs) on partially unfrozen layers.

Model Evaluation:
After training, we evaluated the model on a separate test set to see how well it could classify new, unseen images. The model achieved a high accuracy, indicating that it had learned to recognize the car models effectively.

(c) **Nissan Model Recognition**
Training the Model: The model trained on 7 distinct models (Altima, Armada, Datsun, Maxima, navara, patrol, sunny) process begins by resizing input images to 224×224 pixels to match MobileNetV2's requirements, using a batch size of 32 for efficient processing. The training occurs in two key phases: first, a 60-epoch initial phase with the base model frozen and a learning rate of 1e-4, followed by a 30-epoch fine-tuning phase where the top 30 layers are unfrozen with a reduced learning rate of 1e-5.

Model evaluation: After training, we evaluated the model on a separate test set to see how well it could classify new, unseen images. The model achieved a good accuracy, indicating that it had learned to recognize the car models effectively.

(d) **Toyota Model Recognition**
Training the Model: The model trained on 12 distinct models (toyota-avalon, toyota-camery, toyota-corolla, toyota-fj, toyota-fortuner, toyota-hiace, toyota-hilux,toyota-innova,toyota-landcruiser,toyota-prado,toyota-rav4,toyota-yaris) process begins by resizing input images to 224×224 pixels to meet the input requirement of MobileNetV2's, using a batch size of 32 for efficient processing. The training was divised on two phases: first, a 40-epoch initial phase with the base model frozen and a learning rate of 1e-4, followed by a 10-epoch fine-tuning phase where the top 30 layers are unfrozen with a reduced learning rate of 1e-5.

Model evaluation: After training, we evaluated the model on a separate test set to assess its ability to classify new, unseen images. The model achieved a good accuracy, demonstrating that it had learned to distinguish between the 12 Toyota car models effectively.

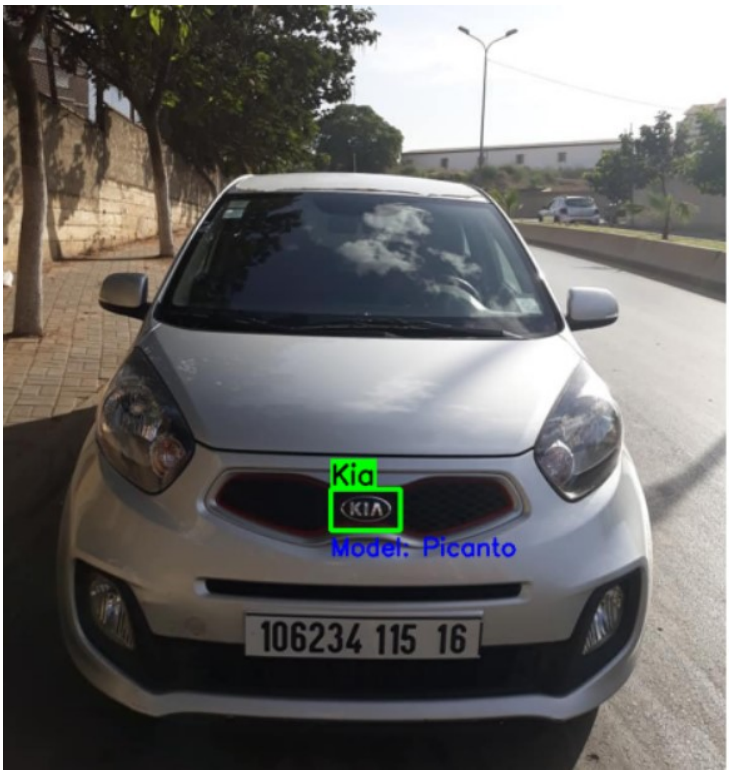


Figure 3.37: An example of applying the selected model for car model prediction.

3.8.3 Color Classification

The proposed CNN model architecture that we used for vehicle color recognition.

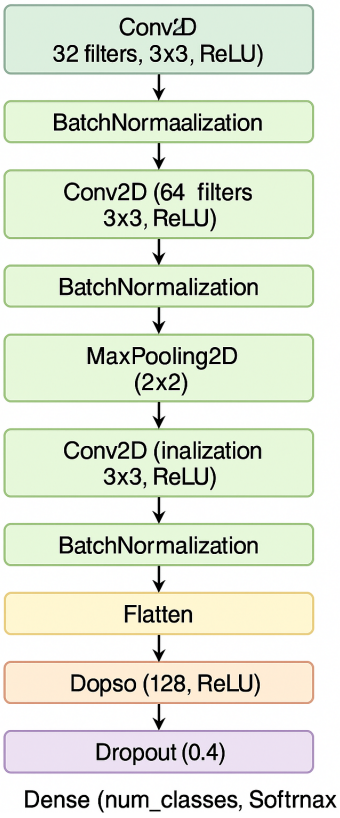


Figure 3.38: The proposed architecture of the CNN model for color classification

The CNN model starts with a **Conv2D layer** with 32 filters (3x3 kernel) and **ReLU activation**, the architecture includes batch normalization to preserve activations and quicken training. Mid-level characteristics are retrieved using a second **Conv2D layer** with 64 filters (3x3), which is once more standardized using batch normalization.

Using spatial downsampling, a **MaxPooling2D** layer (2x2 pool size) then lowers the dimensionality while yet maintaining vital features. Based on the context, the third Conv2D layer which most likely has 128 filters recognizes more sophisticated patterns; BatchNormalization is utilized afterwards.

Utilizing **Dropout** (0.4 rate) for regularization, the 3D feature maps are flatt into a 1D vector (Flatten) and sent to a Dense layer (128 units, ReLU) for high-level feature learning. In essence, the output **Dense layer** with Softmax activation produces class probabilities.

The design is helpful for image categorization programs such vehicle color identification (as shown in the supplied code) because it balances feature extraction and model generalization. The mix of convolutional blocks, normalization, pooling, and dropout guarantees strong learning and lessens overfitting shown in Figure 3.38.

- **Training the model:**
  The CNN model was trained using the VCOR dataset (see Section 3.4). The training data was preprocessed using the ImageDataGenerator class, which applies data augmentation techniques such as rotation, zoom, and horizontal flipping. These techniques enhance the model's generalization ability by exposing it to a wider va-

riety of input conditions.

The model completed 18 training epochs using the Adam optimizer and categorical cross-entropy as the loss function, which are appropriate for multi-class categorization applications. The model evaluates its performance on a separate validation set (10% of the training data is separated into the validation set) and optimizes its weights by reducing the computed loss on the training set over each epoch. Real-time monitoring of loss as well as training and validation accuracy also forms part of the training procedure.

- **Model Evaluation:**
  Following training, the model was evaluated using the evaluate() function on a different test dataset. The model was never exposed to this dataset during training or validation, which guarantees an unbiased assessment of its performance.Figure 3.39 presents a sample of color prediction phase.

Figure 3.39: Sample of color prediction phase

3.8.4 Orientation Classification

For this step, we use the YOLOv8 model for detection. The training is based on the Front and Rear of Car dataset (see Section 3.4), which already includes a data.yaml configuration file. This file specifies the paths to the training, validation, and test image directories. It also defines two target classes to be detected: front and behind. The data.yaml provides all necessary parameters for setting up the training process, including class names and total number of classes.

- **Train model for detection and prediction** The training process for the license plate detection model is initiated using the YOLOv8 architecture. The input images are resized to a resolution of 640x640 pixels, allowing the model to process images at this size. The batch size is set to 8, which means that during each iteration of training, the model processes 8 images together.
  The training is carried out for 50 epochs, where each epoch represents a complete iteration over the entire training dataset. By training the model over multiple epochs, it has the opportunity to learn and improve its performance on the vehicle orientation detection and prediction task. (See Figure 3.40)

- **Model evaluation**
  After training the model, we evaluated its performance on a dedicated set of test images to assess its effectiveness.

Figure 3.40: Sample of orientation prediction with YOLOv8

3.9 Control Access

In this part of the system, a decision is made regarding vehicle access authorization. After the vehicle is detected and its key information is successfully recognized, including license plate number, color, brand, and model, the system proceeds to determine if the vehicle meets the entry criteria. The system first verifies if the license plate matches the Algerian format. If so, it checks the SQLite database for authorization. Access is granted only if the vehicle is both Algerian and listed as authorized; otherwise, entry is denied. Figure 3.41 illustrates this decision process.

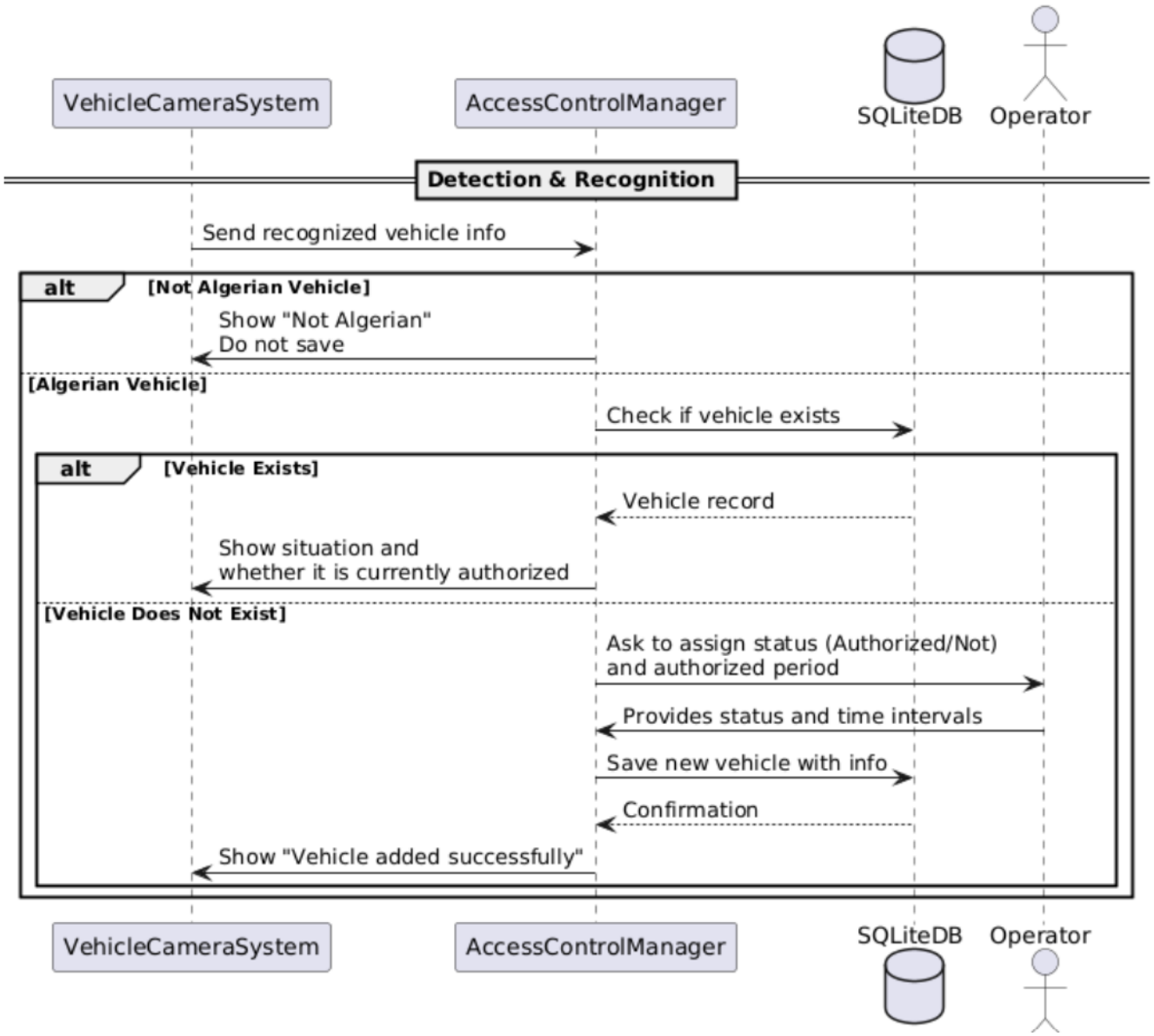


Figure 3.41: Sequence diagram of the proposed system.

This sequence diagram illustrates the three primary scenarios that a vehicle access system can encounter. First, the system merely shows "Not Algerian" and does nothing else when it detects a non-Algerian vehicle. The system determines whether Algerian vehicles are present in the SqliteDB database. It displays their access status if they are registered. If unregistered, it saves the new entry, verifies that "Vehicle added successfully!" and asks the operator to assign a status (Authorized/Not) and time period.

The VehicleCameraSystem (which records data), AccessControlManager, SqliteDB (which stores records), and the Operator are the four main players in the process. When combined, they guarantee that only vehicles with permission can enter.

3.9.1 SQLite Database

SQLite is a lightweight, serverless, self-contained, and embedded relational database management system (RDBMS). Unlike traditional databases like MySQL or PostgreSQL,

SQLite does not require a separate server process and stores the entire database in a single file on disk. It is designed for simplicity, efficiency, and ease of integration into applications.[96]

3.9.2 Verification Workflow

In this part, we present the essential components that detail the proposed access control system.

Key Components in the Verification Process

- VehicleCameraSystem: Captures plate data.
- AccessControlManager: Applies access logic and tracks denials.
- SQLiteDB: Stores vehicle records and access history.
- Operator: Reviews and inputs authorization for unknown vehicles.

Vehicle Access Control Scenarios
The system applies access control by following these verification cases:

1. Non-Algerian Vehicle
   The system analyzes the license plate format. If it doesn't match the Algerian pattern, it displays "Not Algerian" and discards the data; no further action is taken.
2. Registered Algerian Vehicle If the vehicle is Algerian, the system checks the SQLite database:
   - If the vehicle is listed and the current time is within the authorized period: "Access Allowed" is displayed.
   - If the time is outside the permitted range: "Access Denied" is shown, and a denial counter is incremented.
3. Unregistered Algerian Vehicle
   If the vehicle is not found in the database: The system alerts the operator to assign a status (Authorized/Not) and a valid access period.
   Once saved, the system displays "Vehicle Added Successfully".

* Access Control Logic

Definitions and Notation Let:

- P : Detected license plate number.
- $IsAlgerian(P)$: Returns **true** if P matches Algerian license plate format.
- $ExistsInDB(P)$: Returns **true** if P exists in the SQLite database.
- $Authorized(P)$: Boolean; indicates whether the vehicle is authorized to enter.
- $WithinPeriod(P, t)$: Boolean; returns **true** if current time t is within the authorized window for P.

Algorithm 2 License Plate Access Control System

Data: License plate number P, current time now
Result: Access control decision

if *not* $IsAlgerian(P)$ **then**
  display("Not Algerian") exit
end

if $ExistsInDB(P)$ **then**
  if $Authorized(P)$ *and* $WithinPeriod(P, now)$ **then**
    display("Access Allowed")
  end
  else
    $log\_attempt(P)$ **if** $Attempts(P) > 5$ *within last 48 hours* **then**
      $alert\_operator(P)$
    end
    display("Access Denied")
  end
end
else
  $status, time\_range \leftarrow operator\_input()$ $DB.insert(P, status, time\_range)$ display("Vehicle Added Successfully")
end

Example
A delivery truck with an Algerian plate arrives for the first time. The system doesn't find it in the database. The operator grants access only between 10:00 AM–12:00 PM. If the truck returns outside this time or without renewed authorization, access will be denied. This step ensures that only recognized or authorized Algerian vehicles are processed further, maintaining the integrity and security of the access control system.

3.10 Conclusion

In conclusion, this chapter has explained the general design and structure of our project. It showed the main steps we followed and gave a clear idea of how the system works. This part helped us understand how all the different parts of the project are connected and how they work together. The design we presented will guide us as we move on to the next stage. In the following chapter, we will focus on the practical part, where we will show how we built and implemented each part of the system using different tools and methods.

Chapter 4

Implementation

4.1 Introduction

In the preceding chapter, we discussed the conception of our system in detail. In this chapter, we will focus on the development environment and the libraries employed in our system. Additionally, we will provide an overview of the key components of our code and present the obtained results.

4.2 Environment

to realize our application we used tow separate computers, each with the following specifications :

Model Part	**Laptop 1**	**Laptop 2**
Processor	Intel(R) Core(TM) i5-7300U CPU @ 2.60GHz	Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz
RAM	8.00 GB	8.00 GB
System type	64-bit operating system, x64-based processor	64-bit operating system, x64-based processor
Edition	Windows 10 Pro	Windows 10 Fam

Table 4.1: Characteristics of the two computers used

4.2.1 Development environments

In this study, we used the Python programming language along with supporting tools, which are detailed as follows:

A. Python

Python, developed by Guido Van Rossum in the late 1980s, is a powerful, high-level, interpreted, interactive, and object-oriented scripting language suitable for beginner programmers. It supports a wide range of applications, from text processing to web browsers, games, and artificial intelligence. Python's rapid growth is due to its simple syntax, which reads like plain English, making it easy to write complex programs. This paper analyzes

Python's popularity, features, and applications, focusing on web application frameworks used in Python programming.[97].

B. Google Colab

Google Colab, also known as Colaboratory, is a simple and free tool offered by Google Research. With Colab, anybody can develop and execute any Python code using the browser. Machine learning, education, and data analysis are particularly well-suited to this environment. Colab is a setup-free service that hosts Jupyter notebooks and offers unfettered access to computer resources, including GPUs [98].

C. Jupyter notebook

The open-source web application Jupyter Notebook makes it possible to create and share documents with live code, equations, visualizations, and informative text. Jupyter Notebook, which is compatible with several programming languages, including Python, R, and Julia, provides an interactive computational platform that makes it easier to write and run code in an organized and methodical manner.[99]

4.2.2 Libraries

In this study, we used several libraries, which are detailed as follows:

A. Tensorflow

TensorFlow is an open-source machine learning framework created by Google. It offers an extensive collection of resources, libraries, and material to bootstrap and run machine learning models. TensorFlow allows both deep learning and conventional machine learning processes and provides high-level APIs for ease of deployment while still preserving low-level APIs for customization and versatility. TensorFlow is used in a multitude of industries, from computer vision to natural language processing and voice recognition[100].

B. Keras

Keras is a high-level neural networks API, written in Python and ayouvus (which allows operative and CPU or GPU). It is a convenient and easy way to construct and test deep learning models. With Keras, beginners or advanced users can create network with clear and easy-to- understanding. It has multi-backend support and fast experimentation, and it has been widely used in deep learning applications[95].

C. PyTorch

PyTorch is a popular open-source machine learning framework and scientific computing library, known for its versatile approach to building and training deep learning models. Its dynamic computing, linked with NumPy, simplifies mathematical operations and focuses on GPU acceleration, enabling faster training and inference. PyTorch offers a diverse ecosystem with pre-trained models, learning resources, and specialized libraries in fields like computer vision, natural language processing, and audio processing[101].

D. OpenCv

OpenCV is an open-source library widely used for computer vision. The library has robust libraries that provide several functions and algorithms. This includes image processing, video processing, object detection, and feature extraction. OpenCV supports various operating systems and programming languages, but it is most famous for its Python bindings. The library provides an intuitive, easy-to-understand API that developers can use to integrate computer vision into newly developed projects. OpenCV is popular in many fields, including robotics and medical image analysis. It also features comprehensive documentation and pretrained models, which court a large vibrant community[102].

E. Matplotlib

Matplotlib is a library in Python that offers a robust tool for easy visualization of data. It provides many plotting functions and plots that can be fully customized. For example, line plots, scatter plots, bar plots, histograms and many more. It has many elements that can be controlled precisely, like colors, labels, axes and many more, meaning that many modifications can be easily made. It can also easily interact with other Python libraries, thus, making it very popular for data analysis and scientific visualization. Matplotlib provides a versatile interface and documentation, which makes it an excellent tool for quickly and efficiently visualizing the data[103].

F. NumPy

NumPy is a crucial Python library for scientific computing, offering efficient tools for multi-dimensional array manipulation, linear algebra, and random number generation. It integrates with other scientific libraries, making it a popular choice for tasks like data analysis, machine learning, and simulation. Its simplicity, speed, and versatility make it an essential tool for Python numerical computing[104].

G. Gradio

An inventive, open-source Python library called Gradio was made to produce intuitive user interfaces for data applications and machine learning algorithms. Its main goal is to make the process of turning machine learning models into interactive web applications that are simple enough for a wide range of users to understand. Gradio is used by both technical and non-technical users due to its user-friendly interface.[105]

H. PIL (Python Imaging Library)

PIL is a well-known Python library for working with images. It has a lot of features that let you open, edit, and save different types of image files. PIL lets you change the size, crop, rotate, and filter images. It also lets you change the colors, contrast, and brightness of images. The library also lets you do simple things with images, like adding text, drawing shapes, and applying different effects. PIL is a great tool for working with images in Python because it is simple and easy to use. It is widely used in programs that deal with computer vision, graphics, and multimedia processing[106].

4.3 System overview

We dedicate this section to show the interfaces of our system and the function of each of them, as our system has four different interfaces, including the frontend and the main interface, access control interface and finally database interface. with a special focus on the main modules. The figure below shows the front end of our application.

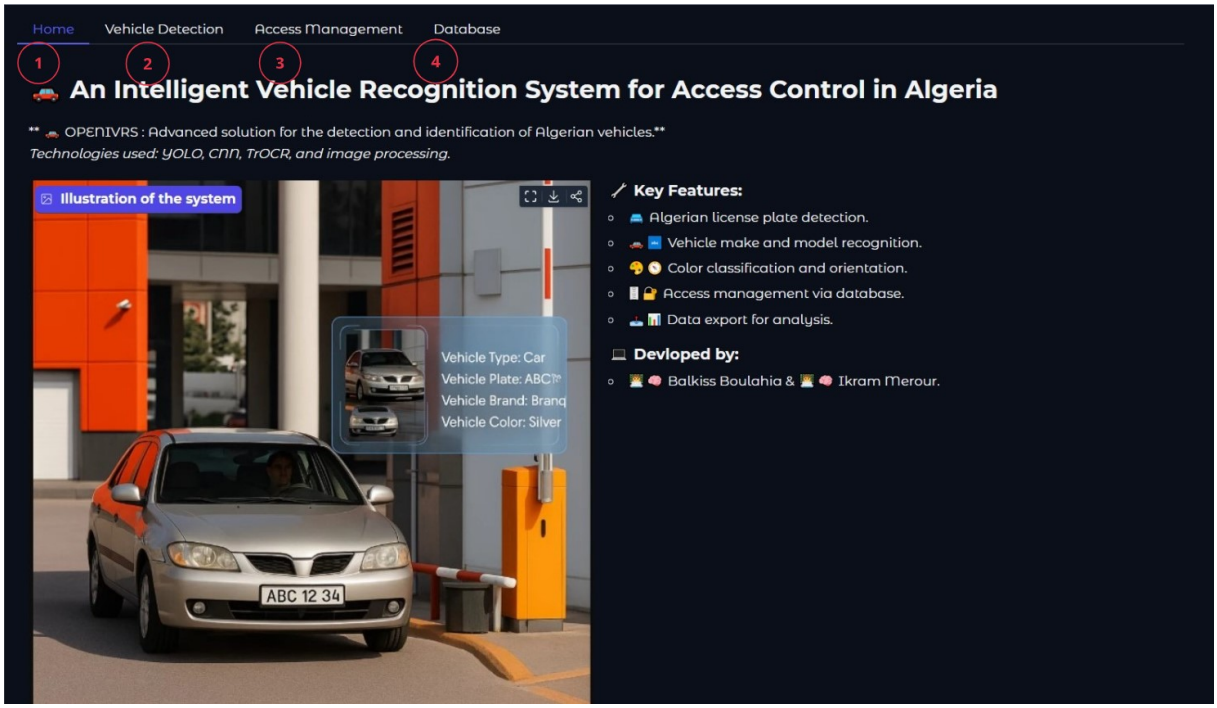


Figure 4.1: The home interface of our system.

After clicking on the "Vehicle Detection", the basic interface shown in the image below appears. (See Figure 30)

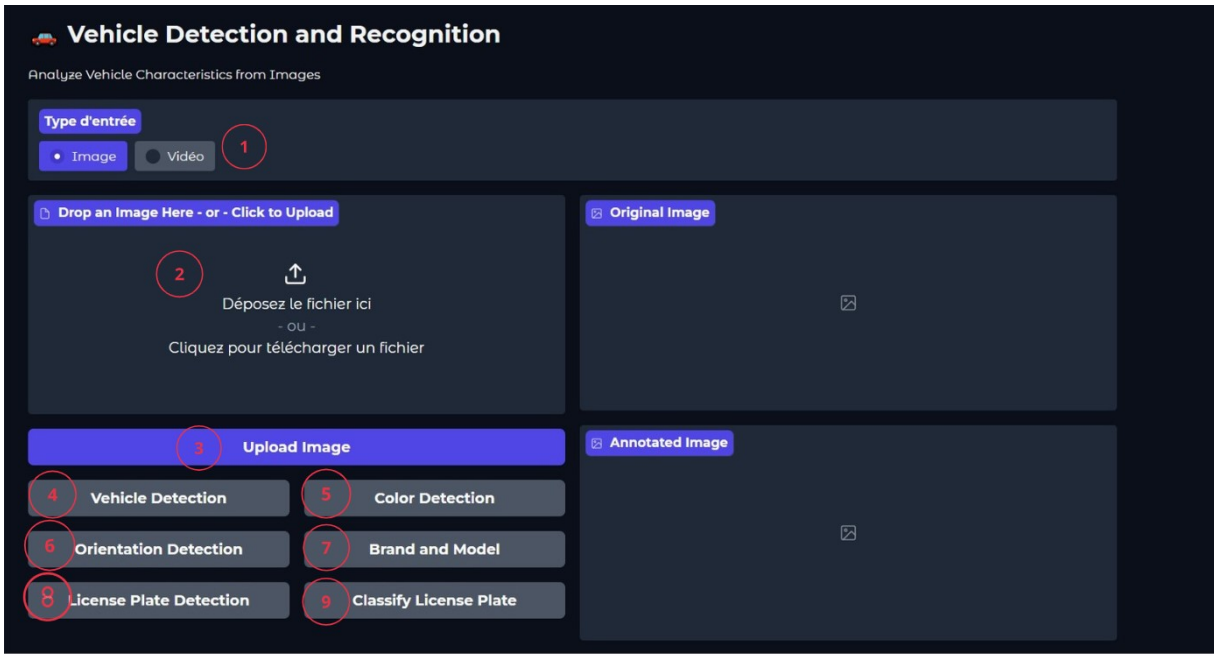


Figure 4.2: Basic interface of our system

The main modules of our application are summarized by the following numbers:

1. The initial radio button allows the user to select the type of input (image or video).
2. The second button enables the user to upload the file (image or video).
3. The third button is designed to perform the task of vehicle detection and type recognition and subsequently draw a bounding box around the identified vehicle.
4. The fourth button is designed to perform the task of color recognition.
5. The fifth button is designed to perform the task of vehicle orientation detection and recognition and subsequently draw a bounding box around the identified vehicle part.
6. The Sixth button is designed to perform the task of logo detection, recognition and model recognition subsequently draw a bounding box around the identified logo.
7. The Seventh button is designed to perform the task of license plate detection and recognition and subsequently draw a bounding box around the identified license plate and also on its characters.
8. The Eighth button is responsible for classify the recognized plate.

After clicking on the ”Access Management”, the interface shown in the image below appears.
(See Figure 31)

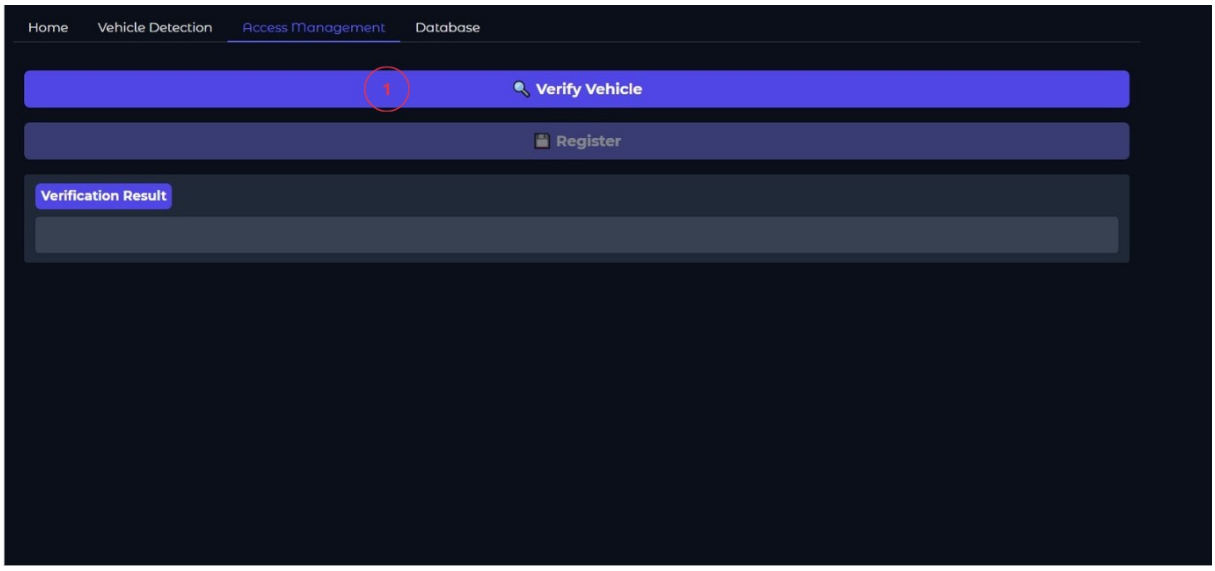


Figure 4.3: Access control interface of our system

(Figure 32) represents the case when the vehicle is not existed in the database.

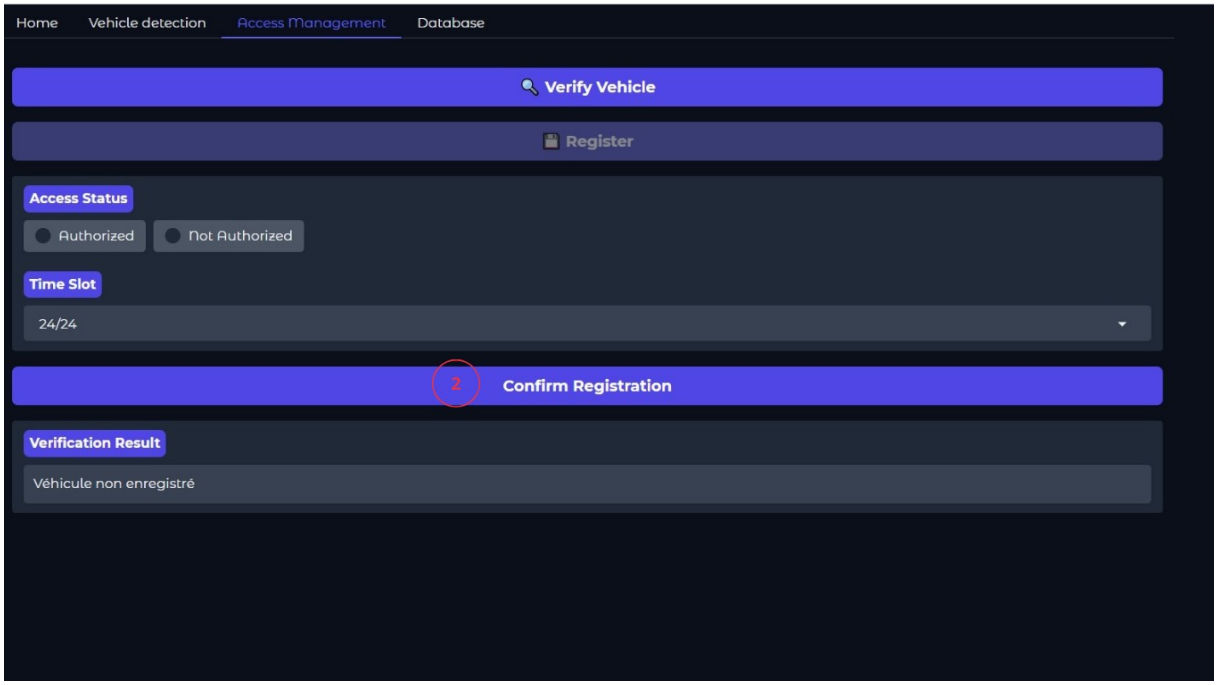


Figure 4.4: Access control interface of our system(new vehicle)

The main modules of this part are summarized by the following numbers:

1. first we click on the button verify vehicle, to check if it already exist or not.
2. second button Confirm registration enables the user to save the information of new vehicles.

After clicking on the ”Database Management”, the interface shown in the image below appears.
(See Figure 33)

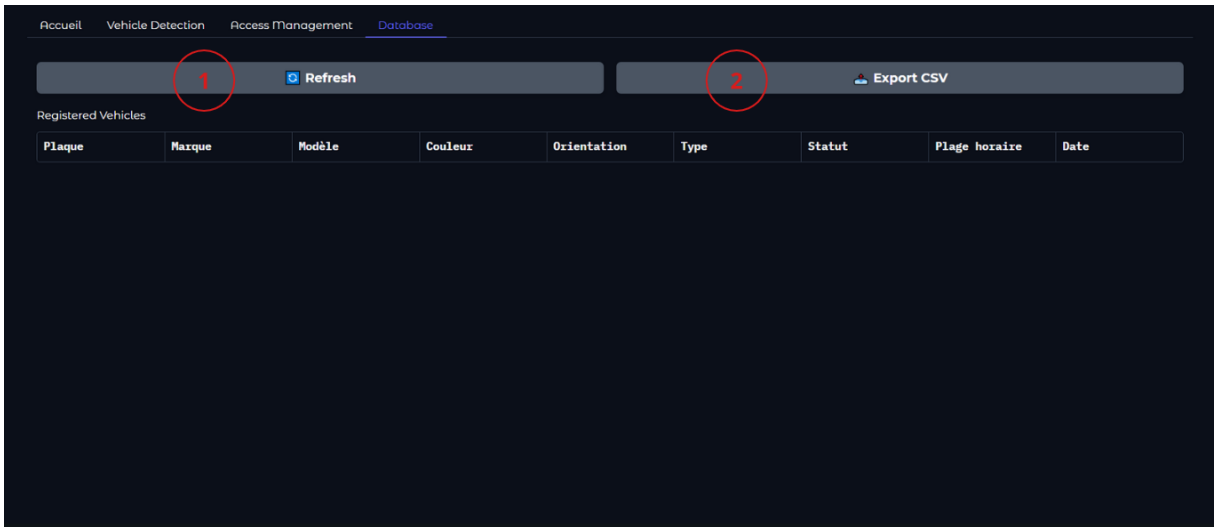


Figure 4.5: Database interface of our system

The main modules of this part are summarized by the following numbers:

1. the button ”refresh” for refresh the database tables.
2. the button ”export CSV” if we want to upload another database file.

4.4 Usage scenario

In this section, we will provide an overview of the working principles of our system. We will describe the different stages involved in vehicle recognition for control access system:

1. To commence the vehicle recognition process, we start by opening a (image/video) file using the command ”Upload file” This allows us to access the desired images and videos and utilize it for further analysis and detection procedures. As shown in Figure 34

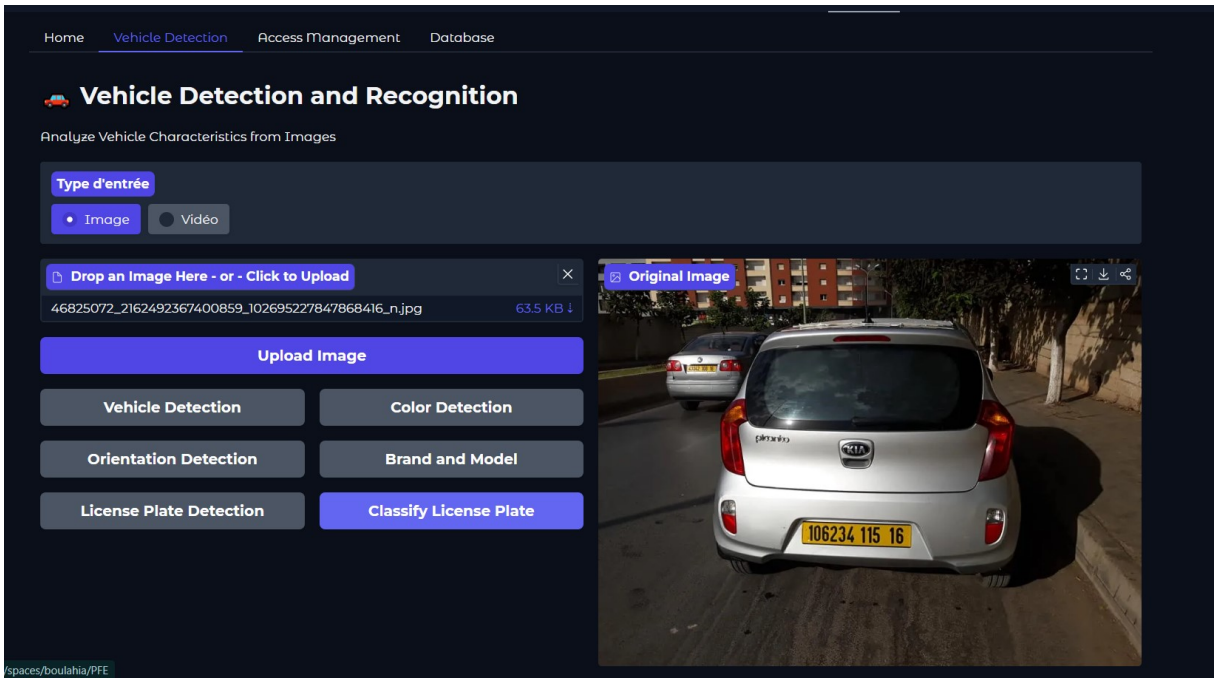


Figure 4.6: Input selection (image or video)

If the input type is a video we should extract the frames before we start the vehicle identification process, as shown in Figure 35

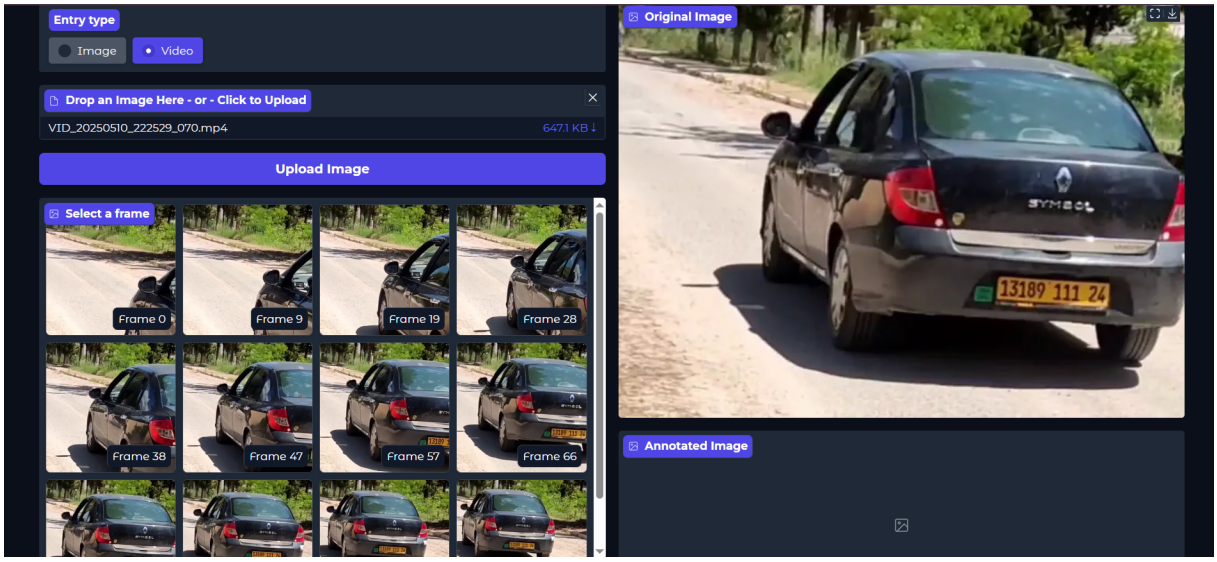


Figure 4.7: Frames extraction from input video

2. after opening the file successfully. When we click the "vehicle detection" button, the process of identifying the car in the opening picture or video begins, the YOLOv8 model which has already been trained on a dataset allows for precise and effective vehicle detection. The outcome displayed in Figure 36

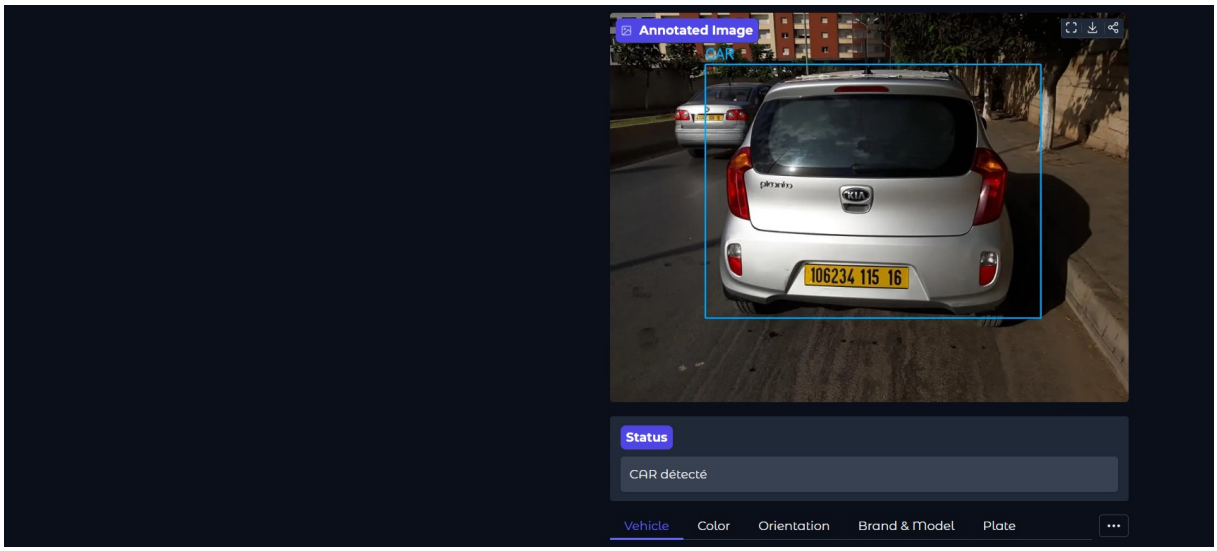


Figure 4.8: Vehicle detection & Recognition

3. When we click the "Color detection" button, the process of prediction the car color begins, the CNN model which has already been trained on a dataset allows for precise and effective color prediction. The outcome displayed in Figure 37

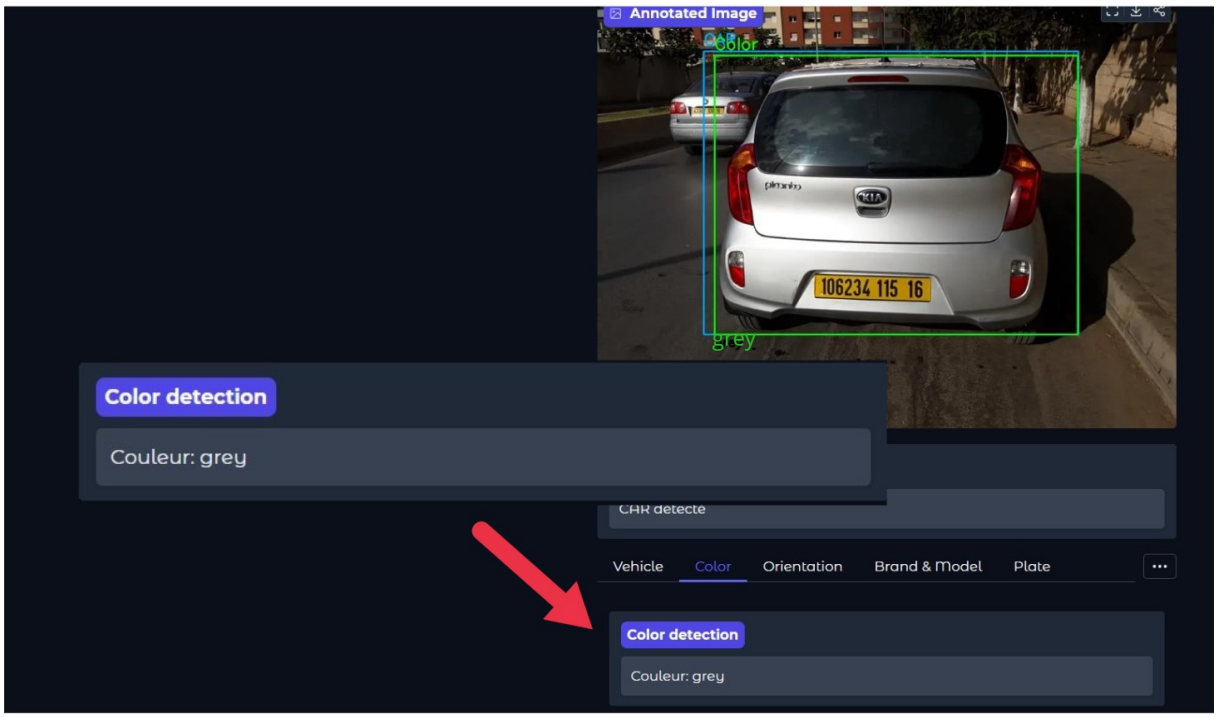


Figure 4.9: Color prediction

4. The button "Direction detection" start the process of identifying the vehicle orientation, the YOLOv8 model which has already been trained on a dataset allows for precise and effective vehicle detection. The outcome displayed in Figure 38

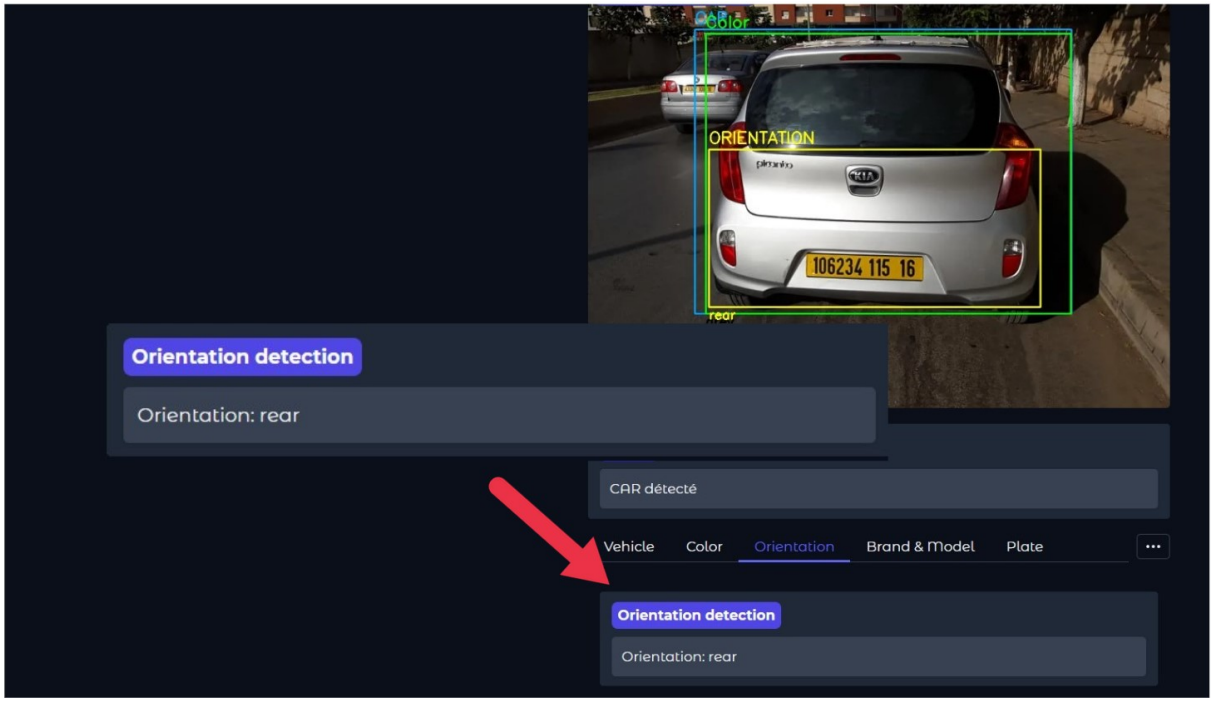


Figure 4.10: Orientation detection & recognition

5. When we click the "Brand and model" button, the process of prediction the brand and model begins, the CNN model which has already been trained on a dataset allows for precise and effective prediction. The outcome displayed in Figure 52

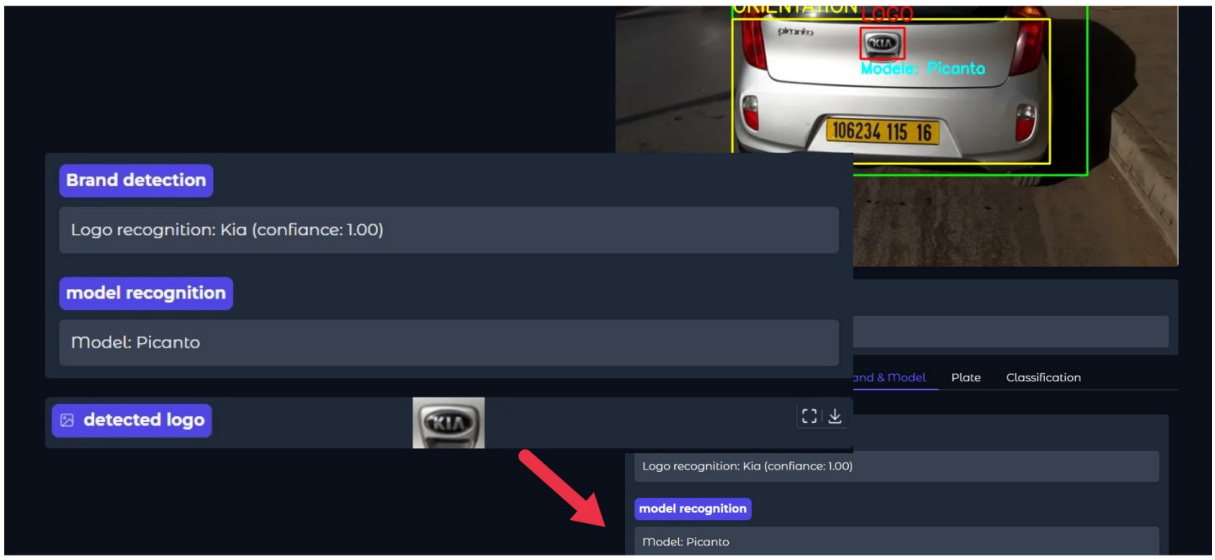


Figure 4.11: Make & Model Recognition

6. By selecting the "License plate detection" button, we initiate the process of detecting the license plate within the opened image and also recognize its number with TrOCR than show the list of the recognized characters. The result shown in Figure 40

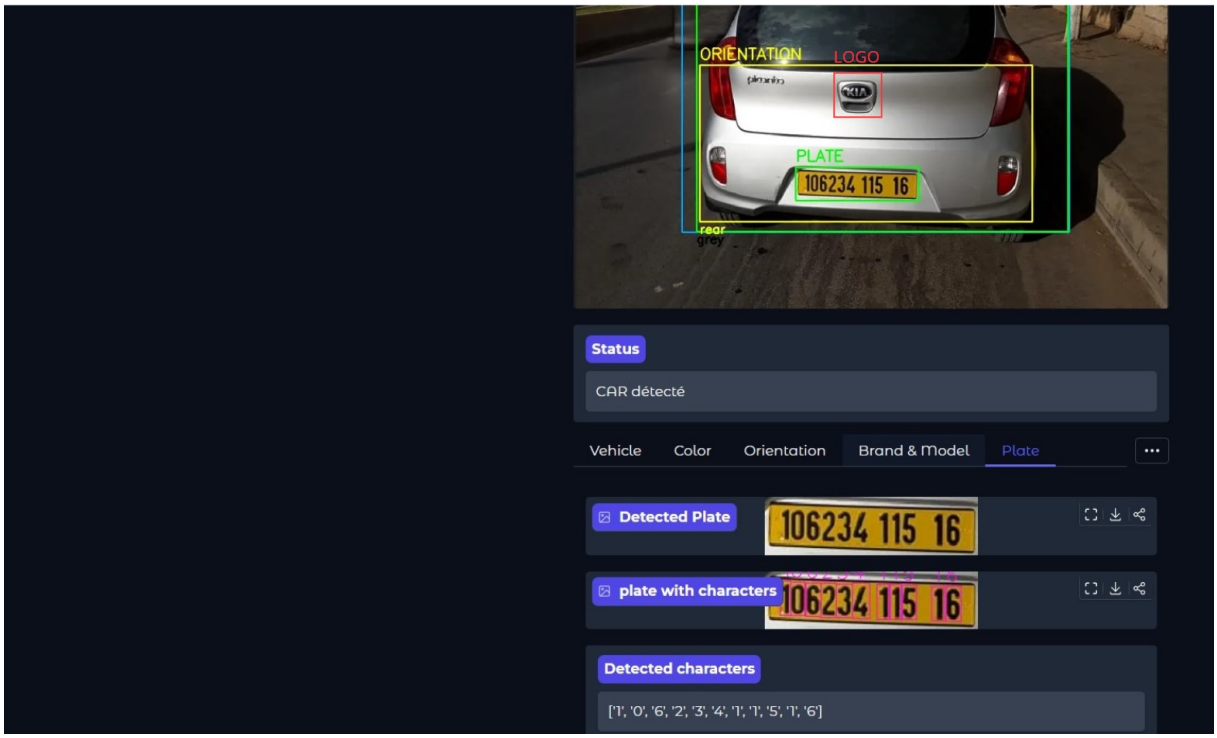


Figure 4.12: License plate detection and character recognition

7. A license plate classification procedure is carried out via the "classify plate" button. It classifies the final two digits, the year of the license plate, and the type of vehicle after reading license plate numbers from the list of characters in each step. As seen in Figure 41, the extracted data is subsequently shown, offering information about the license plate such as its translation, year, kind, and remaining characters. Additionally, Figure 42 shows an illustration of a non-Algerian plate.

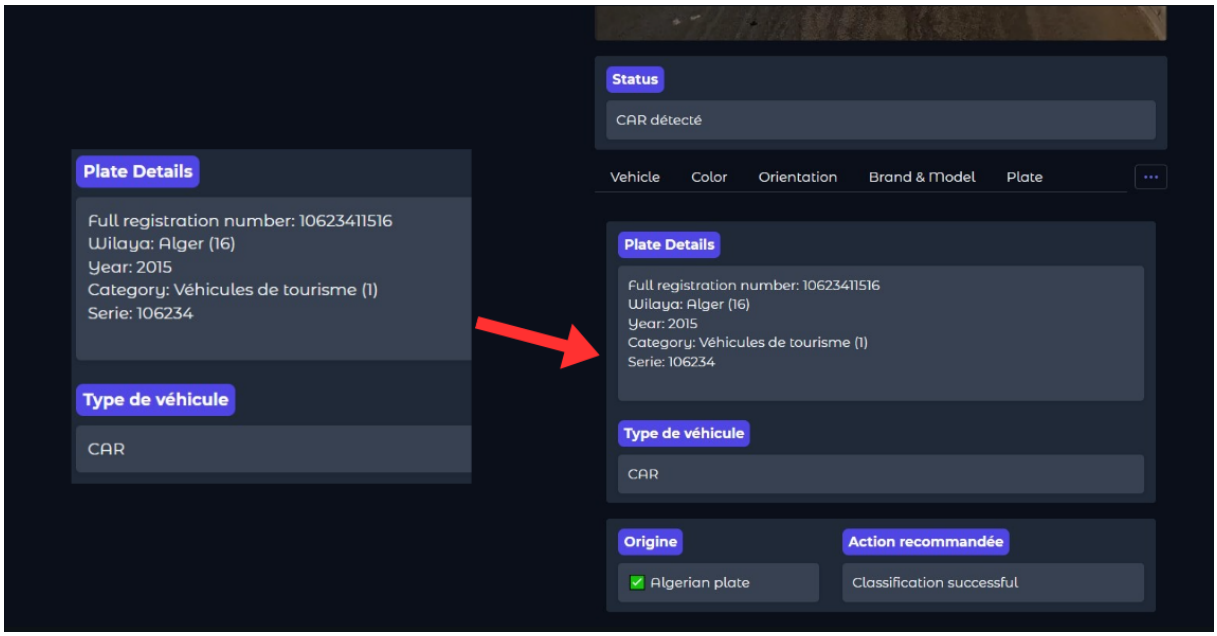


Figure 4.13: Character Classification.

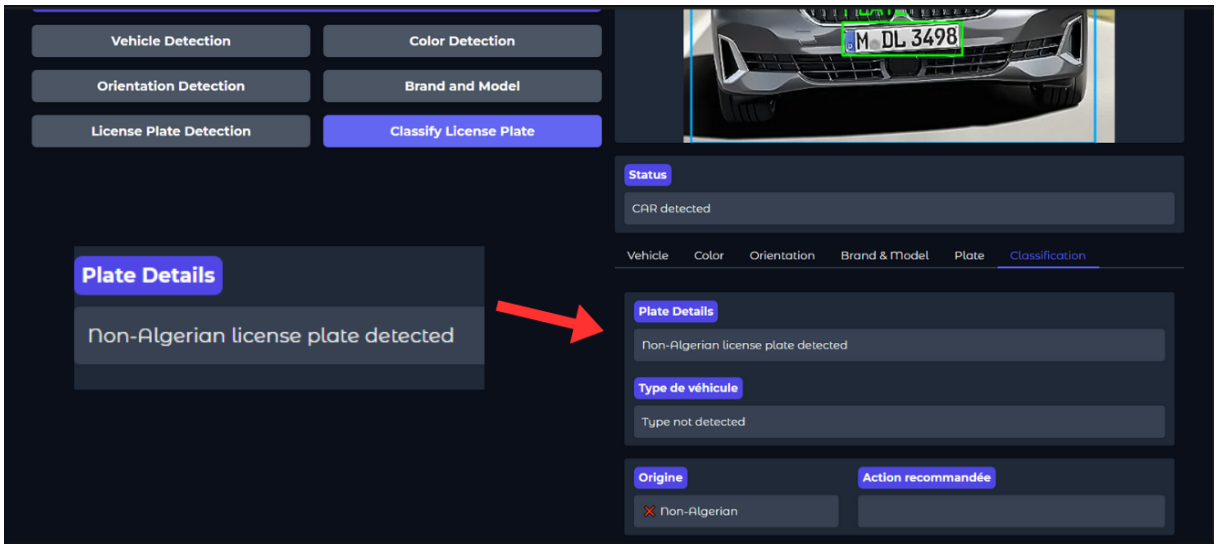


Figure 4.14: Example of non-Algerian plate.

8. The button ”verify vehicle” start the process of checking if the vehicle already exist or not. The outcome displayed in Figure 52. and after how to save it

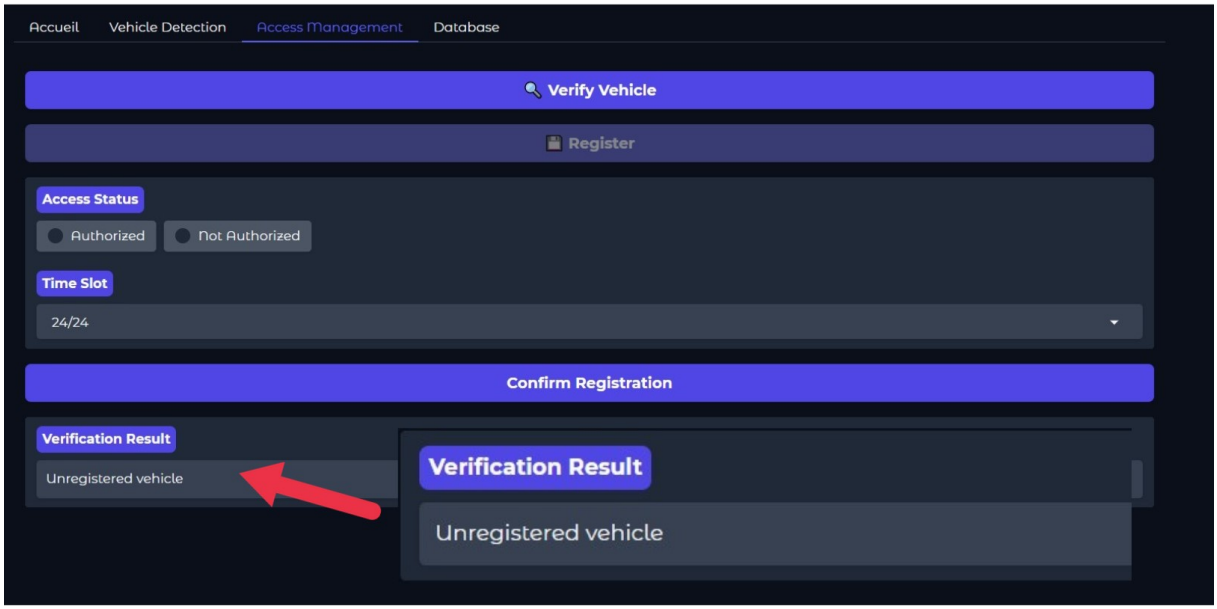


Figure 4.15: Vehicle existence verification

after that we select the Access status and the time slot and save it.

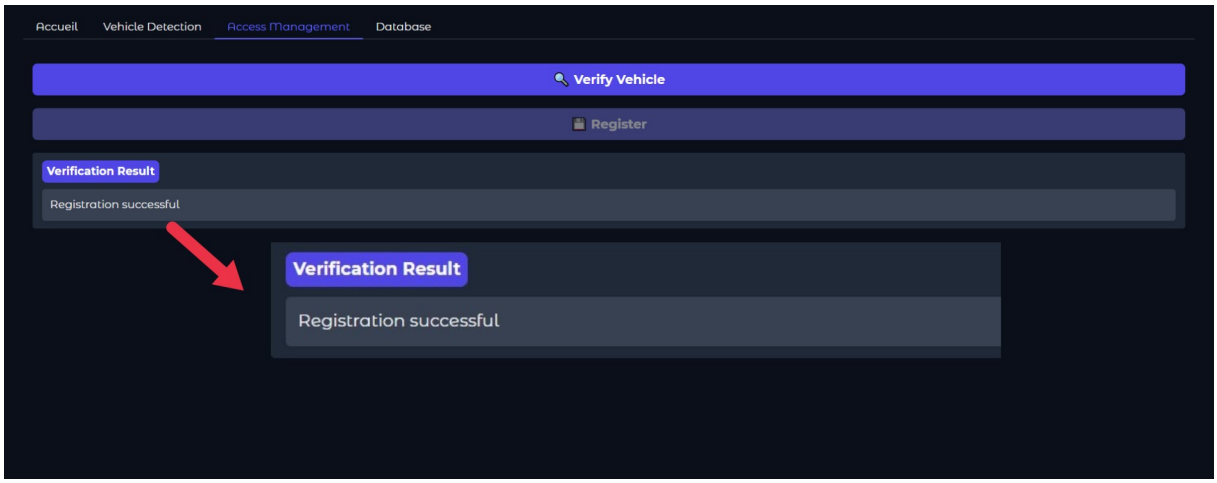


Figure 4.16: Save new vehicle entry

9. The button "refresh" start the process of showing the list of vehicles and their information. The outcome displayed in Figure 45.

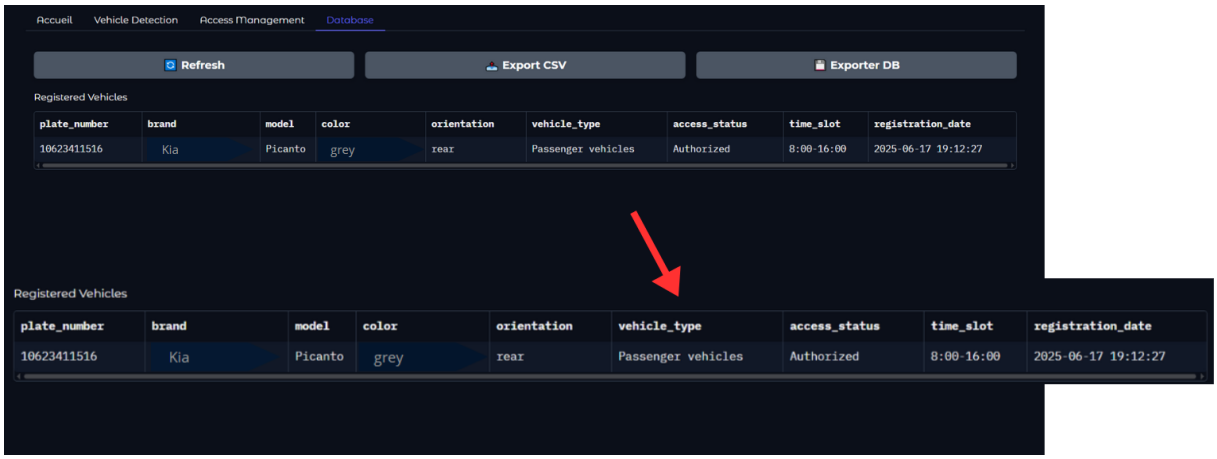


Figure 4.17: Vehicle list display

4.5 Model Performance and Analysis

In this section, we detail the results, which consist of three parts: presentation of results, analysis, and comparison of the models among themselves and with other related works that share similar features

4.5.1 presentation of results

Here, we present the results of each phase. To evaluate these models, we use the following components:

Evaluation indicators When evaluating the performance of our models, several indicators are commonly used. These indicators help measure the accuracy and efficiency of the model.[107]

- **Precision :** (metrics/precision(B)) The precision metric calculates the ratio of accurately predicted bounding boxes to all expected bounding boxes for an object. It shows how well the algorithm reduces false positives.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.1}$$

  Where:

  - TP (**True Positives**): Number of correct positive class predictions
  - FP (**False Positives**): Number of errors where negatives are predicted as positives

- **Recall :** (metrics/recall(B)) The percentage of accurately predicted bounding boxes for objects relative to the total number of ground truth bounding boxes is known as recall. It shows how well the algorithm can identify items and reduce false negatives.

$$\text{Recall} = \frac{TN}{TN + FP} \tag{4.2}$$

  Where:
  TN (**True Negatives**): The model does not predict the label and is not part of the ground truth.

- **Average Precision (AP):** A popular statistic that blends recall with precision is called AP. By taking precision-recall curves into account, it assesses the model's overall accuracy across several item categories. The average AP value across several item categories is called Mean Average Precision, or mAP.53

$$\text{Average Precision (AP)} = \int_{r=0}^{1} p(r)\, dr \tag{4.3}$$

- **Loss:** indicates the degree to which the model's predictions and the actual outcomes agree. It calculates how much the actual values depart from the projected values. Many machine learning techniques aim to reduce this loss

A. vehicle detection & Recognition

In our system, we used YOLOv8n, a miniature version of YOLOv8, for detecting vehicles in both images and videos. The detailed steps were explained in the previous chapter. The training result of the model is shown in Table [4.2] and Figure [4.18].

Table 4.2: The training result statistics of Vehicle detection and Recognition.

Precision	**Recall**	**mAP50**
95%	92%	97%

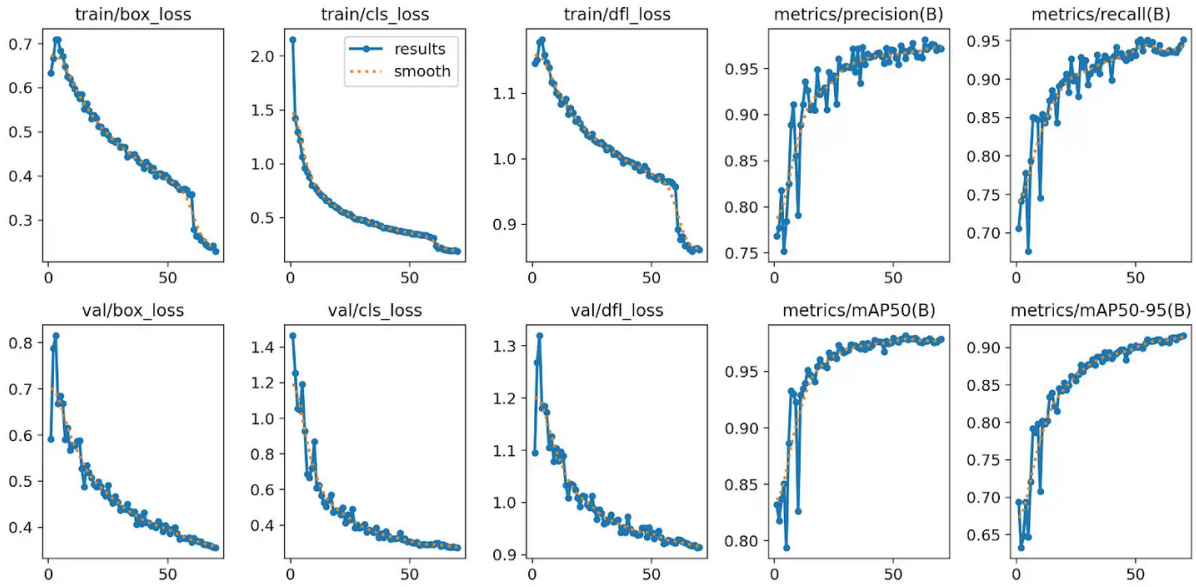


Figure 4.18: The training results of YOLOv8 of vehicle detection and Recognition.

To evaluate the performance of our YOLOv8n model, we tested it on new images that were not used during training. The model successfully detected vehicles accurately across various scenarios.

B. Plate detection & Recognition

1- Plate Detection

In this phase, we have used YOLOv8s.for detecting number plates in images. we explained the steps in the previous chapter . The training result of the model are shown in Table[4.3] and Figure[4.19].

Table 4.3: The training result statistics(plate detection).

Precision	**Recall**	**mAP50**
97 %	99 %	99 %

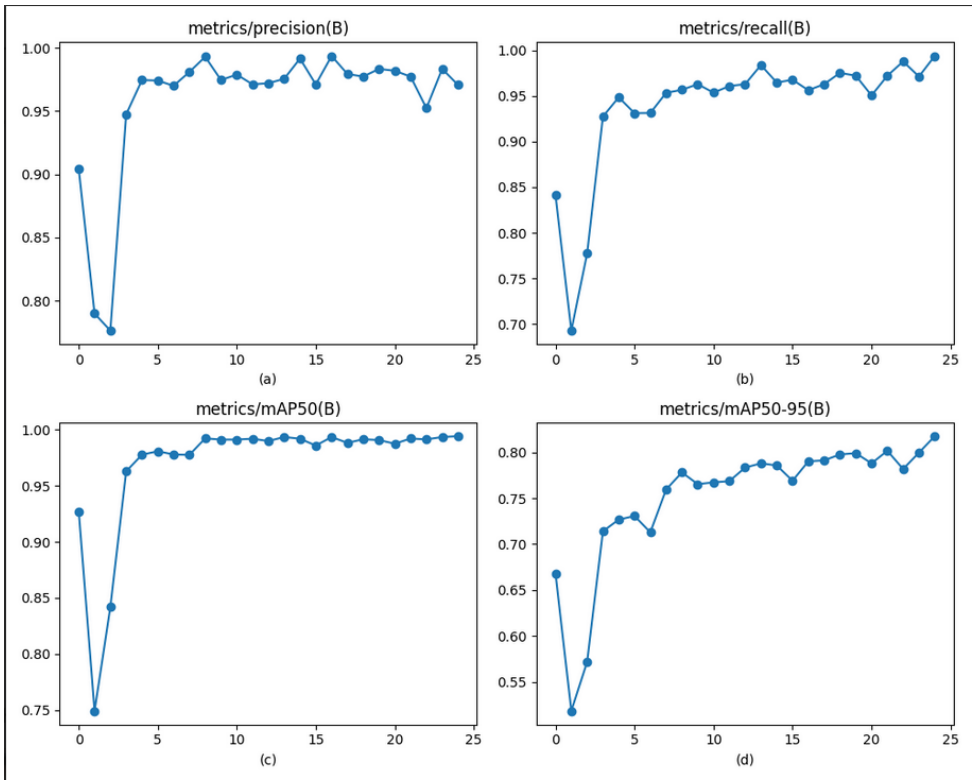


Figure 4.19: The training results of YOLOv8.

From (a) and (c) in Figure 4.19, the accuracy and mapping convergence of the model to 0.9,which means that the model detection accuracy is high enough, In Figure (b) also he recall convergence converges to 0.9, indicating that the target can be detected completely.

2- Characters Detection

In this phase, we have used YOLOv8s. a miniature version of YOLOv8, for detecting number plates in images. we explained the steps in the previous chapter. The training result of the model are shown in Table[4.4] and Figure[4.20].

Table 4.4: The training result statistics(characters detection).

Precision	**Recall**	**mAP50**
95 %	93%	97 %

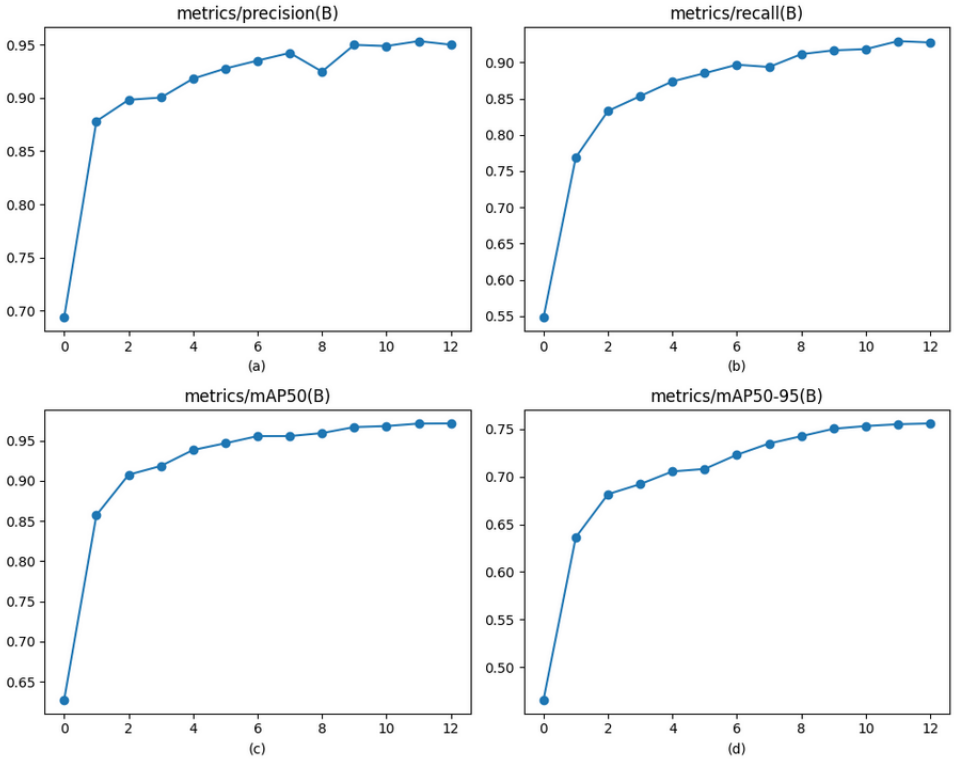


Figure 4.20: The training results of YOLOv8.

From (a) and (c) in Figure[4.20], the accuracy and mapping convergence of the model to 0.9,which means that the model detection accuracy is high enough, In Figure (b) also he recall convergence converges to 0.9 ,indicating that the target can be detected completely.

3- Characters Recognition

For the character recognition phase, we tested our system on a set of 50 extracted during the license plate detection stage. To evaluate the results, we used a ground truth file that contains the actual license plate numbers for each image. The predictions obtained from our OCR system were then compared to these ground truth values. The results are presented in the table below[4.5]

Table 4.5: Statistics of character recognition results.

Precision	**Recall**
86%	85.71%

C. Logo, model detection & Recognition

1. **Model detection**
   In our system we have used YOLOv8s. a miniature version of YOLOv8, for detecting number plates in images. we explained the steps in the previous chapter . The training result of the model is shown in Table[4.6] and Figure[4.21].

Table 4.6: The training result statistics(Logo detection).

Precision	Recall	mAP50
86%	83 %	87%

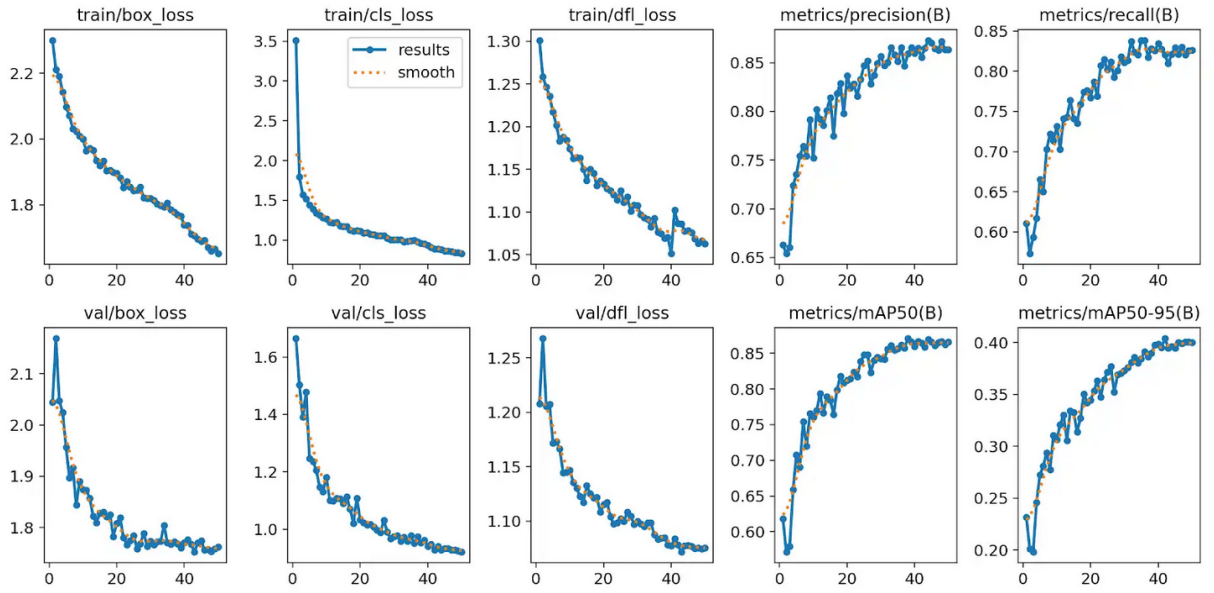


Figure 4.21: The training results of YOLOv8.

b. Logo Recognition

The accuracy and loss curve of the CNN model is shown in Figure Figure 4.22.

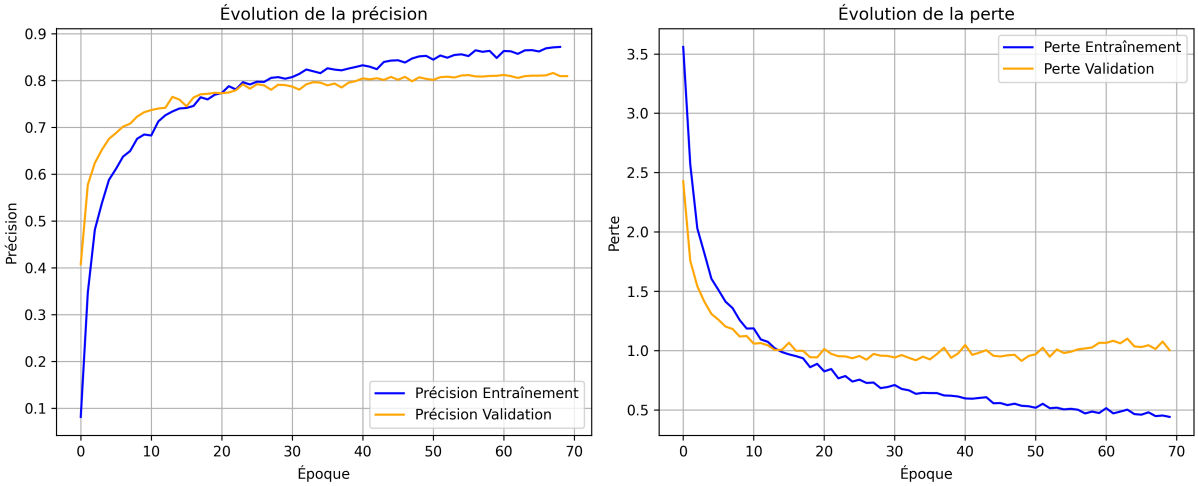


Figure 4.22: Accuracy & loss Curves of the CNN model

The model successfully reduced the difference between the expected and actual values, as evidenced by its validation loss of 1.0030. This implies that, on the whole, the model's predictions closely match the actual data. Additionally, the model accurately classified a sizable number of the validation samples, as indicated by the validation accuracy of 0.8096. These findings suggest that the model is highly accurate in differentiating between

classes or categories. Overall, the evaluation results highlight the model's potential and effectiveness in resolving the issue at hand, confirming its capacity to generate precise forecasts and enhancing the overall success of your study.

B. Models Recognition

In the model recognition phase, we have four car models that were recognized using MobileNet integrated with CNN.

1. **KIA Model Recognition**
   The model achieved a validation loss of 0.4021, the model accurately classified the validation samples, as indicated by the validation accuracy of 0.8592. These findings suggest that the model is highly accurate in differentiating between classes or categories. The accuracy and loss curve is shown in Figure 4.23

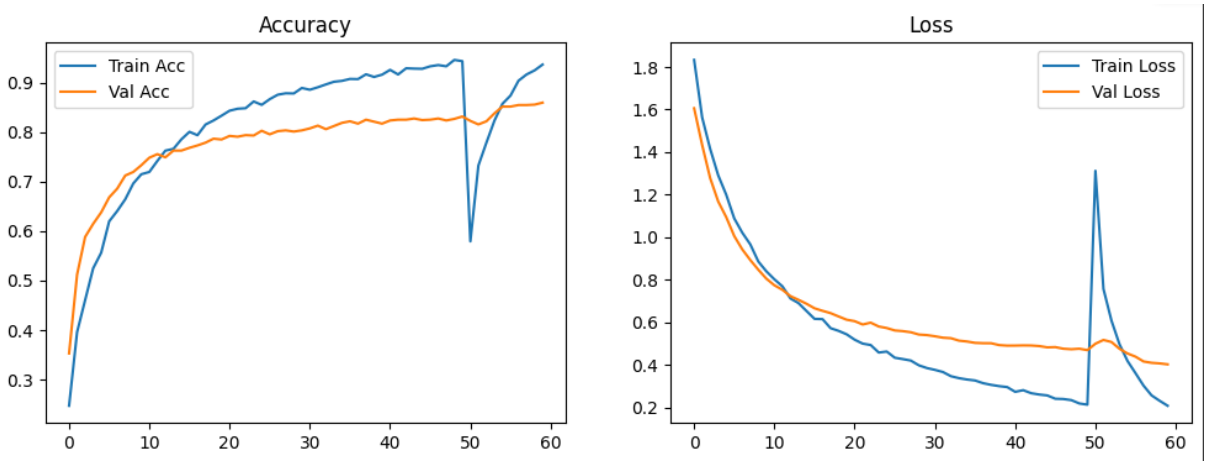


Figure 4.23: Accuracy & Loss Curves of KIA model

2. **Chevrolet Model Recognition**
   The model achieved a validation loss of 0.2768, the model accurately classified the validation samples, as indicated by the validation accuracy of 0.9282. These findings suggest that the model is highly accurate in differentiating between classes or categories.

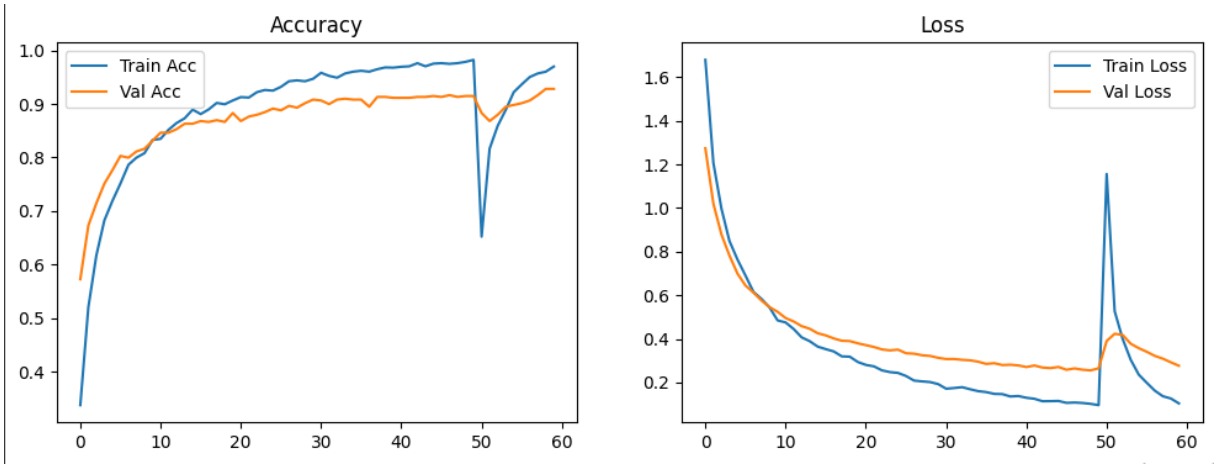


Figure 4.24: Accuracy & Loss Curves of Chevrolet model

3. **Nissan Model Recognition**
   The model achieved a validation loss of 0.3931, the model accurately classified the validation samples, as indicated by the validation accuracy of 0.9063. These findings suggest that the model is highly accurate in differentiating between classes or categories. The accuracy and loss curve is shown in Figure 4.25

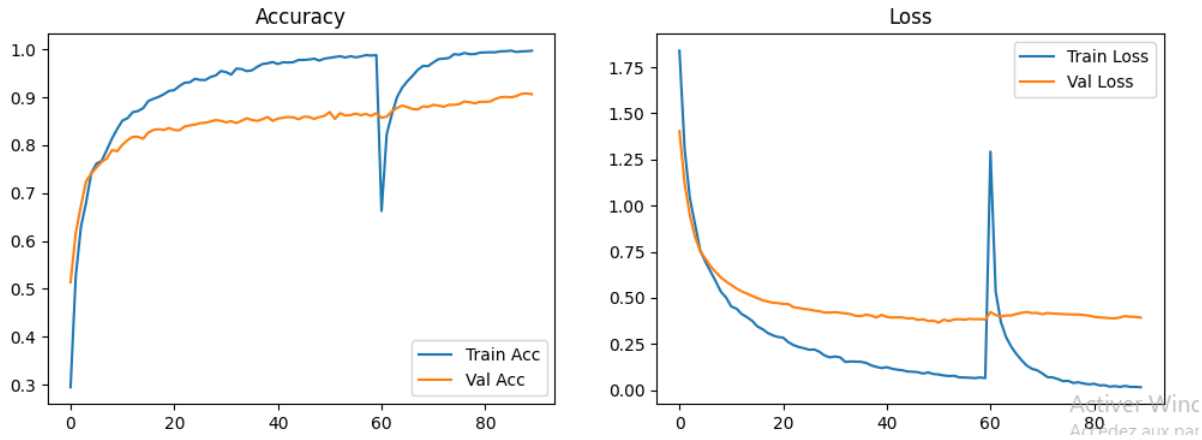


Figure 4.25: Accuracy & Loss Curves of Nissan model

4. **Toyota Model Recognition**
   The model obtained a validation loss of 0.7896 , demonstrating effective classification of the validation samples, which is reflected in a validation accuracy of 0.9063. These results indicate that the model is very proficient in distinguishing between different classes or categories. The accuracy and loss curve is displayed in Figure 4.26

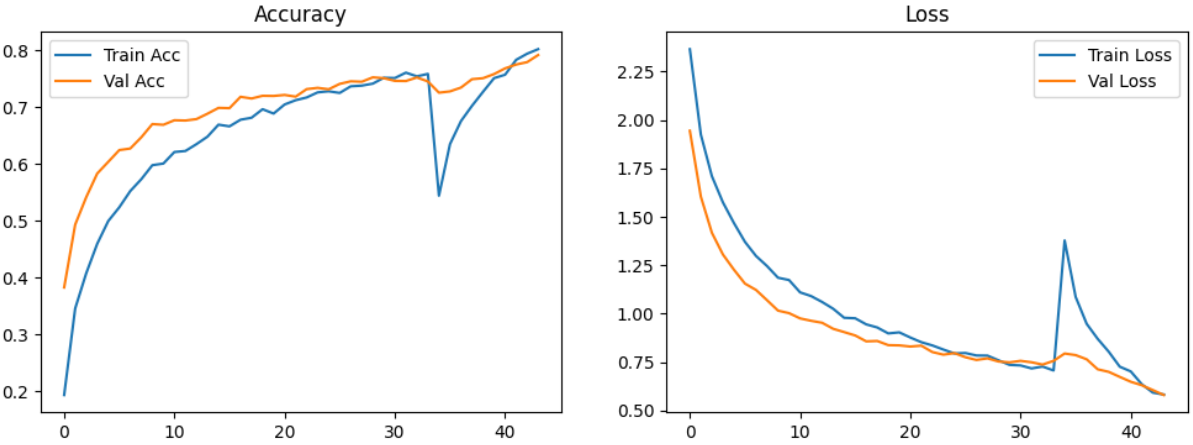


Figure 4.26: Accuracy & Loss Curves of Toyota model

It is important to note that the datasets used for training: **Kia, Chevrolet, Nissan, and Toyota** contain 6139, 5097, 5 950, and 12150 images respectively. Each vehicle model within these datasets includes between 200 and 500 images, which is considered a relatively small amount for deep learning tasks. This limited data per class can make it difficult for the model to distinguish between visually similar vehicle models, often leading to confusion in classification.

D. Color prediction

The CNN model successfully reduced the difference between the expected and actual values, as evidenced by its validation loss of 0.5160. This implies that, on the whole, the model's predictions closely match the actual data. Additionally, the model accurately classified a sizable number of the validation samples, as indicated by the validation accuracy of 0.8775. These findings suggest that the model is highly accurate in differentiating between color classes. Overall, the evaluation results highlight the model's potential and effectiveness in resolving the issue at hand, confirming its capacity to generate precise forecasts and enhancing the overall success of your study.The accuracy and loss curve is shown in Figure[4.27]

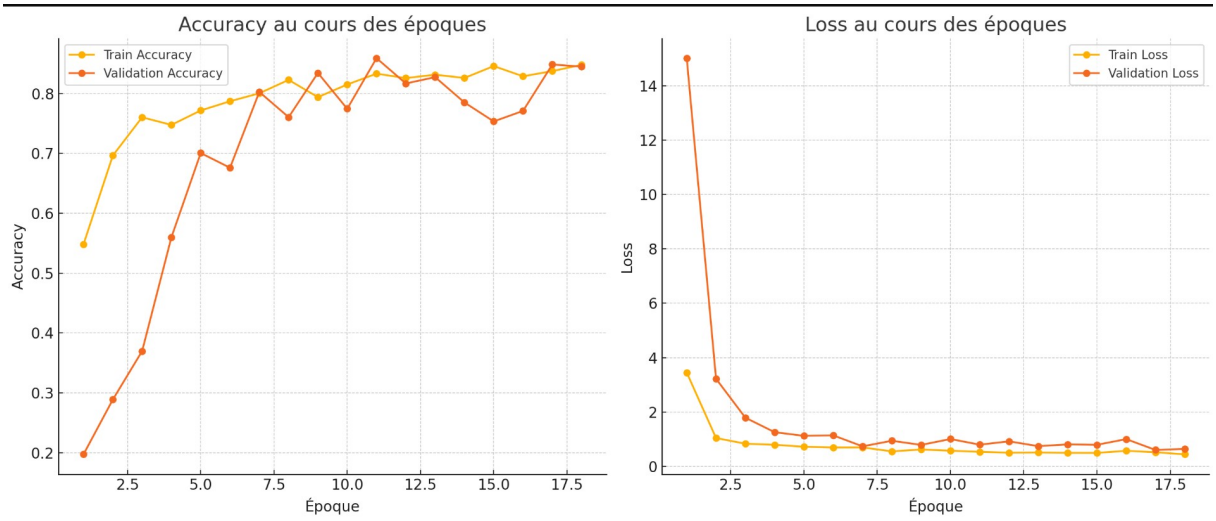


Figure 4.27: Accuracy & loss Curves of Color

E. Direction detection & Recognition

For the direction recognition task, we used the YOLOv8 model to identify the orientation of the vehicle accurately. The training result of the model is shown in Table[4.7] and Figure[4.28]

Table 4.7: The training result statistics of Direction Detection.

Precision	**Recall**	**mAP50**
95%	93%	98%

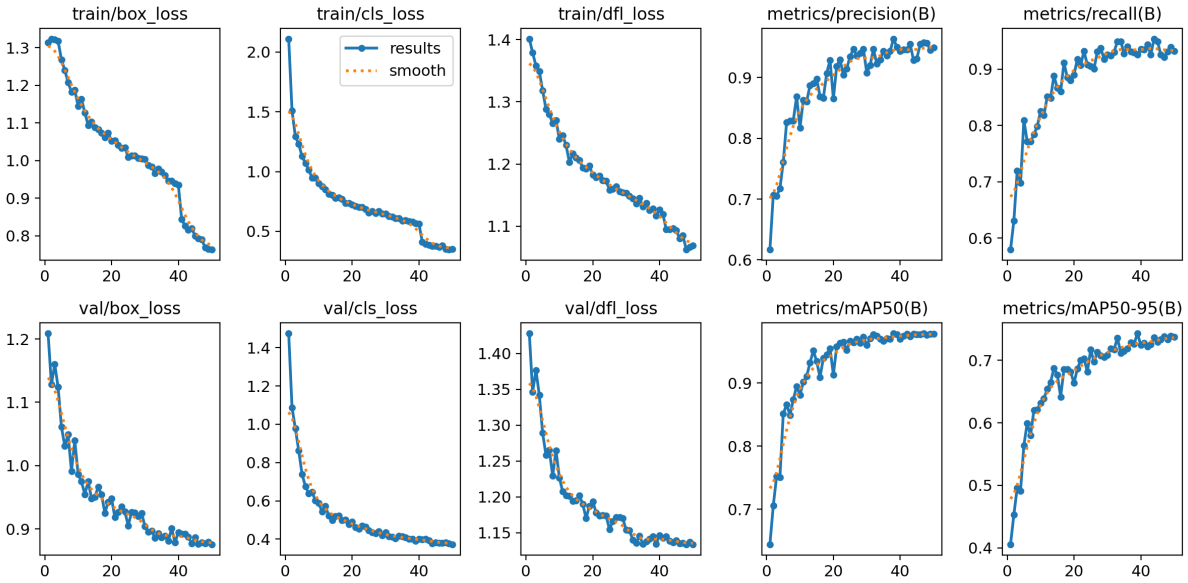


Figure 4.28: The training results of YOLOv8 of Direction Recognition.

From Figure[4.28], the accuracy and mapping convergence of the model to 0.95, which means that the model detection accuracy is high enough, also the recall convergence converges to 0.93,indicating that the target can be detected completely.

4.6 Discussion

In this section, we provide a result analysis based on the outcomes of the models presented in this study. The evaluation is divided into four main models: YOLOv8, CNN1, CNN2(MobileNet), and CNN3. The table below summarizes the performance results for each phase of our system.

Table 4.8: Comparison Table of System Models

Phase	**Model**	**Dataset**	**Precision**	**Recall**
Vehicle Detection	YOLOv8	Vehicle Dataset for YOLO dataset	95%	92%
Plate Detection	YOLOv8	AT HACKATHON	97%	99%
Characters Detection	YOLOv8	ALKO1	95%	93%
Characters ecognition	OCR	50 originally selected images	86%%	85.71%
Logo Detection	YOLOv8	Cars Logo Dataset	86%	83%
Logo Recognition	CNN1	Car Logos Dataset	82%	81%
Vehicle model Recognition	CNN2 (MobileNet)	Car Make, Model, and Generation	Kia: 85% Nissan: 93% Chevrolet: 89% Toyota: 79%	Kia: 84% Nissan: 92% Chevrolet: 88.5% Toyota: 81%
Color Recognition	CNN3	VCOR	83%	80%
Orientation Prediction	YOLOv8	Front and Rear of Car	95%	93%

From the results, it is evident that our system performs well in the stages of detection and recognition of vehicle and the other stage of vehicle identification, particularly in normal cases. The high precision and recall values indicate that the system successfully identifies and classifies all the vehicle feauters with a high level of accuracy. This implies that the system is effective in detecting and recognizing vehicle feauters under typical conditions, showcasing its robustness and reliability.

These positive results indicate the successful performance of our system in handling various scenarios and provide confidence in its ability to accurately process vehicle feauters information.

To summarize, we can say that our system solved many problems and handle many obstacles, which can be described as follows :

- Our system has the capability to detect various types vehicles, logos and license plates color and orientation with high accuracy and efficiency.
- Our system excels in both vehicle detection and recognition, even in challenging conditions such as skew problems and varying illuminations.
- Our system demonstrates high precision in vehicle, brand, color, orientation and license plate detection, ensuring accurate localization of logos and license plates. Additionally, it achieves exceptional accuracy in license plate recognition, enabling reliable identification of numeric characters.

However, like any developed system, we encountered several challenges that are still open in our system, which are presented in the following points:

- The training process for all the vehicles feauters requires a substantial amount of time to complete.
- In the characters detection phase, we encounter challenges where certain characters may not be correctly detected due to character connections or the presence of stickers or other non-relevant characters on the license plate.
- In the vehicle Model recognition phase, we encounter challenges where certain models are similar to each other,so we have difficulty to predict the right vehicle models.

4.7 Conclusion

In this chapter, we present the implementation details of our vehicle identification system. For vehicle detection and recognition task we use YOLOv8 model, which has shown remarkable performance in object detection and recognition. For license plate processing, we combine YOLOv8's precise plate detection with specialized OCR character detection and recognition. The make/model recognition employs a hierarchical approach: First, YOLOv8 performs logo detection, and then passes cropped logo regions to a CNN classifier for brand identification. This dual-stage design addresses the challenge of inter-brand similarity. For model recognition, we implement brand-specific CNN classifiers trained on distinct vehicle Models, we encountered significant difficulties in distinguishing between visually similar vehicle models. and finally for color direction directiOn we use YOLOv8, which has shown remarkable performance in object detection and recognition task.

General Conclusion

The intelligent vehicle recognition system for access control is increasingly valuable due to its ability to enhance security, streamline vehicle management, improve parking systems, and support smart city initiatives. Therefore, developed systems are highly needed today, given the rapid advancement of AI technologies globally. Specifically in Algeria, we notice the absence of such advanced systems, where access control is still handled manually and AI is not yet integrated into fields such as parking management, boom barriers, or blacklisted vehicle detection, where most operations are still performed by humans.

In this thesis, we offer a comprehensive solution that goes beyond simple license plate recognition, providing full vehicle recognition adapted to the specific requirements of access control in the country. Our system integrates different well-known models, where the YOLOv8 algorithm is dedicated to many tasks including vehicle detection and recognition, logo identification, orientation estimation, and license plate detection. Customized CNN models are also employed for accurate brand, model, and color recognition, along with OCR processing specifically adapted for Algerian plates. Different datasets were collected, including global ones and others specifically focused on Algerian vehicles.

This integrated approach enables the system to achieve high precision in identifying and processing multiple vehicle attributes simultaneously. Through extensive testing, our solution has demonstrated strong performance in vehicle recognition tasks, showing high levels of accuracy, efficiency, and robustness, making it suitable for various security and access control applications, including government sites, corporate zones, restricted areas, hotels, residences, and universities across Algeria.

Looking ahead, we have identified several key directions to further enhance our system:

- Expanding our training dataset to include more Algerian vehicle models and plate variations to improve recognition accuracy; Optimizing our detection and recognition algorithms to handle challenging conditions more effectively.
- Enhancing our real-time processing capabilities through algorithm refinement and hardware optimization to support high-throughput access points.
- and Developing more sophisticated anomaly detection features to identify suspicious vehicles automatically.

By pursuing these improvements, we aim to establish our system as the benchmark for intelligent vehicle recognition and access control solutions in Algeria, offering unprecedented security and operational efficiency for critical infrastructure and facilities.

Bibliography

[1] Talhi Haroun. "Conception et implémentation d'un système de contrôle d'accès pour les péages autoroutiers à l'aide de la reconnaissance des plaques d'immatriculation." In: *DSpace - Université du 8 Mai 1945 Guelma* (2024). URL: http://dspace.univ-guelma.dz/jspui/handle/123456789/16451.

[2] Nico Ricky Adytia and Gede Putra Kusuma. "Indonesian License Plate Detection and Identification using Deep Learning." In: *International Journal of Emerging Technology and Advanced Engineering* 11 (07 2021). DOI: 10.46338/ijetae0721_01. URL: www.ijetae.com.

[3] Reza Fuad Rachmadi and I Ketut Eddy Purnama. "Vehicle Color Recognition using Convolutional Neural Network." In: *Department of Multimedia and Networking Engineering, Institut Teknologi Sepuluh Nopember* (2018). Email: fuad@its.ac.id, ketut@te.its.ac.id.

[4] GikBrain. *Reconocimiento de Color del Vehículo*. Tech. rep. GikBrain, 2023. URL: https://gikbrain.com/reconocimiento-color-vehiculo.html.

[5] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. 3rd. Pearson Education, 2008.

[6] M. Emami and M. Fathy. "Color Image Segmentation Using K-Means Clustering and Region Growing." In: *Proceedings of the International Conference on Machine Vision*. 2005.

[7] M. Z. Zia et al. "Vehicle type, make and model recognition using visual logos." In: *Machine Vision and Applications* 30.5 (2019), pp. 837–850.

[8] Cheng-Kai Lin et al. "Vehicle Make and Model Recognition Using Deep Learning: A Survey." In: *Sensors* 22.21 (2022), p. 8439. DOI: 10.3390/s22218439.

[9] *Vehicle Detection and Counting System using OpenCV*. Online Article. 2023. URL: https://www.analyticsvidhya.com/blog/2021/12/vehicle-detection-and-counting-system-using-opencv/.

[10] Alireza Ferdowsi, Mohd Azlishah Rahim, Norlida Rosli, et al. "Vehicle make and model recognition using machine vision techniques: A review." In: *Sensors* 20.4 (2020), p. 1033. DOI: 10.3390/s20041033.

[11] COCKPIT. "Algerian license plate standards (January 2023 update)." In: *Cockpit - Across of Cars* (2023). Accessed on February 18, 2025. URL: https://www.cockpitdz.com/en/post/algerian-license-plate-standards.

[12] Sunil Karan et al. "Vehicle re-identification: an efficient baseline using triplet embedding." In: *IEEE Transactions on Intelligent Transportation Systems* 21.4 (2020), pp. 1364–1376.

[13] AT HACKATON. *AT-Hackathon Dataset*. RoboFlow Universe. Accessed: 2025-06-10. 2024. URL: https://universe.roboflow.com/at-hackaton/at-hackathon/dataset/1.

[14] Wei Li et al. "Vehicle detection and orientation estimation using a multi-task deep convolutional neural network." In: *Remote Sensing* 9.11 (2017), p. 1170.

[15] Yoonho Kim, Hyeonjun Park, and Sanghoon Park. "Rotate YOLO: Real-Time Arbitrary-Oriented Object Detection for Aerial Imagery Using YOLO." In: *Journal of Institute of Control, Robotics and Systems* 27.7 (2021), pp. 532–538. DOI: 10.5302/J.ICROS.2021.21.0006.

[16] Taoufik Saidani and Yamen El TOUATI. "A vehicle plate recognition system based on deep learning algorithms." In: *Multimedia Tools and Applications* 80 (Dec. 2021), pp. 1–12.

[17] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 580–587.

[18] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." In: *European Conference on Computer Vision (ECCV)*. 2014, pp. 346–361.

[19] Ross Girshick et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 580–587.

[20] Jamil Fayyad et al. "Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review." In: *Sensors* 20 (July 2020), p. 4220. DOI: 10.3390/s20154220.

[21] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916.

[22] Peiyuan Jiang et al. "A Review of Yolo Algorithm Developments." In: *Procedia Computer Science* 199 (2022), pp. 1066–1073.

[23] Shubham, Bhomik Sharma, and soumyadip. "Mastering All YOLO Models from YOLOv1 to YOLOv12: Papers Explained (2025)." In: *LearnOpenCV* (Dec. 2023). URL: https://learnopencv.com/mastering-all-yolo-models/.

[24] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[25] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement." In: *arXiv preprint arXiv:1804.02767* (2018).

[26] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection." In: *arXiv preprint arXiv:2004.10934* (2020).

[27] Shahnaj Parvin, Liton Rozario, and Md Islam. "Vehicle Number Plate Detection and Recognition Techniques: A Review." In: *Advances in Science, Technology and Engineering Systems Journal* 6 (Mar. 2021), pp. 423–438. DOI: 10.25046/aj060249.

[28] MathWorks. *What Is a Convolutional Neural Network?* Accessed: 2025-02-06. 2025. URL: https://www.mathworks.com/discovery/convolutional-neural-network.html?utm_source=chatgpt.com.

[29] Wikipedia. *Convolutional Neural Network*. Accessed: 2025-02-06. 2025. URL: https://en.wikipedia.org/wiki/Convolutional_neural_network?utm_source=chatgpt.com.

[30] IBM. *What are Convolutional Neural Networks?* Accessed: 2025-02-06. 2025. URL: https://www.ibm.com/think/topics/convolutional-neural-networks.

[31] Amazon Web Services. *What is RNN? - Recurrent Neural Networks Explained*. Accessed: 2025-02-06. 2025. URL: https://aws.amazon.com/what-is/recurrent-neural-network/.

[32] Weijiang Feng et al. "Audio visual speech recognition with multimodal recurrent neural networks." In: May 2017, pp. 681–688. DOI: 10.1109/IJCNN.2017.7965918.

[33] N. Saranya V. Gnanaprakash N. Kanthimathi. "Automatic number plate recognition using deep learning." In: *IOP Conference Series: Materials Science and Engineering* 1084 (2021), p. 012027. DOI: 10.1088/1757-899X/1084/1/012027.

[34] Carl Writes. *Optical Character Recognition (OCR)*. Accessed: 2025-06-10. 2024.

[35] Avigilon. *License Plate Recognition System*. Accessed: 2025-02-06. 2025. URL: https://www.avigilon.com/license-plate-recognition-solutions.

[36] Admin. "Top Use Cases of Automatic Number Plate Recognition (ANPR) Technology of AI-Powered Video Analytics." In: *JARVIS Blog* (June 2022). Accessed: 2025-02-06. URL: https://www.staqu.com/top-use-cases-of-automatic-number-plate-recognitionanpr-technology-of-ai-powered-video-analytics/.

[37] C. -F.Hsieh et al. "Automatic Vehicle License Plate Recognition Based on YOLO v4 for Smart Parking Management System." In: *2022 IEEE 11th Global Conference on Consumer Electronics (GCCE)*. Osaka, Japan: IEEE, Oct. 2022, pp. 905–906. DOI: 10.1109/GCCE56475.2022.10014165.

[38] Wikipedia contributors. *Vehicle registration plates of Algeria — Wikipedia, The Free Encyclopedia*. Accessed: 2025-05-08. 2025. URL: https://en.wikipedia.org/wiki/Vehicle_registration_plates_of_Algeria.

[39] Jai karan sing Pankaj Sharma and Jai karan sing. "Challenges and Overview of License Plate Character Segmentation." In: *International Journal of Computer Sciences and Engineering* 3 (2 2015).

[40] *Algerian Vehicle Registration Plates Specifications*. Image file. Extracted details: OVAL code: DZ; Series: Passenger, Diplomatic. Accessed: 2025-05-08 at [current time 11:49 am]. 2025. URL: https://licenseplatemania.com/landenpaginas/algerije.htm.

[41] Jithmi Shashirangana et al. "Automated License Plate Recognition: A Survey on Methods and Techniques." In: *IEEE Access* 9 (2020), pp. 11203–11225. DOI: 10.1109/ACCESS.2020.3047929.

[42] Vinita Nishad, Vishal Kumar, and Poornima Mittal. "License Plate Detection by Edge Detection Algorithm and Smart Exposition with IoT." In: Aug. 2021, pp. 1–6. DOI: 10.1109/SASM51857.2021.9841216.

[43] Author(s). "License Plate Detection and Recognition Using Faster R-CNN and CNN-RNN Hybrid Model." In: *arXiv preprint* X.Y (2024), Z. URL: https://arxiv.org/html/2412.12572v1.

[44] Author(s). "An Efficient Vehicle Number Recognition System for Automated Toll Collection." In: *International Journal of Intelligent Systems and Applications in Engineering (IJISAE)* (2024). URL: https://www.ijisae.org/index.php/IJISAE/article/view/4548.

[45] Yuh-Rau Wang, Wei-Hung Lin, and Shi-Jinn Horng. "A sliding window technique for efficient license plate localization based on discrete wavelet transform." In: *Expert Systems with Applications* 38.4 (2011), pp. 3142–3146. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2010.08.106. URL: https://www.sciencedirect.com/science/article/pii/S0957417410009024.

[46] Hong-ke et al. "A new approach of the vehicle license plate location." In: USA: IEEE Computer Society, 2005. ISBN: 0769524052. DOI: 10.1109/PDCAT.2005.24. URL: https://doi.org/10.1109/PDCAT.2005.24.

[47] R. Zunino and Stefano Rovetta. "Vector quantization for license-plate location and image coding." In: *Industrial Electronics, IEEE Transactions on* 47 (Mar. 2000), pp. 159–167. DOI: 10.1109/41.824138.

[48] N Eswar and D Gowri Shankar Reddy. "Morphological operation based vehicle number plate detection." In: *International Journal of Engineering Research And* 9.2 (2020), pp. 428–433.

[49] Zied Selmi, Mohamed Ben Halima, and Adel M. Alimi. "Deep Learning System for Automatic License Plate Detection and Recognition." In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 1105–1110. DOI: 10.1109/ICDAR.2017.187.

[50] Li Zou et al. "License Plate Detection with Shallow and Deep CNNs in Complex Environments." In: *Complexity* 2018 (2018), pp. 1–6. DOI: 10.1155/2018/7984653. URL: https://doi.org/10.1155/2018/7984653.

[51] Sergio Montazzolli Silva and Claudio Rosito Jung. "License Plate Detection and Recognition in Unconstrained Scenarios." In: *Proceedings of the European conference on computer vision(ECCV)* (2018), pp. 580–596.

[52] Weidong Min et al. "New approach to vehicle license plate location based on new model YOLO-L and plate pre-identification." In: *IET Image Processing* 13 (7 2019), pp. 1041–1049. DOI: 10.1049/iet-ipr.2018.6449. URL: https://digital-library.theiet.org/doi/abs/10.1049/iet-ipr.2018.6449.

[53] Lele Xie et al. "A New CNN-Based Method for Multi-Directional Car License Plate Detection." In: *IEEE Transactions on Intelligent Transportation Systems* PP (Jan. 2018), pp. 1–11. DOI: 10.1109/TITS.2017.2784093.

[54] Maglad Khalid, Mohamad Dzulkifli, and N A. "Saudian Car License Plate Number Detection and Recognition Using Morphological Operation and RBF Neural Network." In: 5 (Dec. 2011), pp. 1780–1786.

[55] Yoon Youngwoo et al. "Blob Extraction based Character Segmentation Method for Automatic License Plate Recognition System." In: Oct. 2011, pp. 2192–2196. DOI: 10.1109/ICSMC.2011.6084002.

[56] Qiao Shuang et al. “Research of improving the accuracy of license plate character segmentation.” In: 2010. DOI: 10.1109/FCST.2010.75.

[57] Munna Nur et al. “Intelligent system for vehicles number plate detection and recognition using convolutional neural networks.” In: *Technologies* 9 (Jan. 2021), p. 9. DOI: 10.3390/technologies9010009.

[58] Rahman M M Shaifur et al. “Bangla license plate recognition using convolutional neural networks (CNN).” In: (Sept. 2018). DOI: 10.48550/arXiv.1809.00905.

[59] Yujie Liu et al. “Convolutional neural networks-based intelligent recognition of Chinese license plates.” In: *Soft Computing* 22.7 (2018), pp. 2403–2419. ISSN: 1432-7643.

[60] Hui Li and Chunhua Shen. “Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs.” In: (2016). URL: https://arxiv.org/abs/1601.05610.

[61] Teik Koon Cheang, Yong Shean Chong, and Yong Haur Tay. “Segmentation-Free Vehicle License Plate Recognition Using ConvNet-RNN.” In: (2017). arXiv: 1701.06439. URL: https://arxiv.org/abs/1701.06439.

[62] Wang H, Zhang Y, and Liu J. “Real-time Automatic License Plate Recognition Using CNN-RNN Hybrid Model.” In: *arXiv preprint arXiv:2011.14936* (2020).

[63] Masahiro Daibo. “Toroidal Vector-Potential Transformer.” In: *2017 Eleventh International Conference on Sensing Technology (ICST)*. IEEE. 2017, pp. 1–4. DOI: 10.1109/ICSensT.2017.8304422.

[64] Victor Nascimento Ribeiro and Nina S. T. Hirata. “Efficient License Plate Recognition in Videos Using Visual Rhythm and Accumulative Line Analysis.” In: (2025).

[65] Saif Ur Rehman et al. “An efficient approach for vehicle number plate recognition in Pakistan.” In: *The Open Artificial Intelligence Journal* 06 (May 2020), pp. 12–21. DOI: 10.2174/1874061802006010012.

[66] Irina Pustokhina et al. “Automatic vehicle license plate recognition using optimal k-means with convolutional neural network for intelligent transportation systems.” In: *IEEE Access* PP (May 2020), pp. 92907–92917. DOI: 10.1109/ACCESS.2020.2993008.

[67] Grzegorz Wieczorek et al. “Vehicle Detection and Recognition Approach in Multi-Scale Traffic Monitoring System via Graph-Based Data Optimization.” In: *Sensors* 23.3 (2023). ISSN: 1424-8220. DOI: 10.3390/s23031731. URL: https://www.mdpi.com/1424-8220/23/3/1731.

[68] Yu Peng et al. “Vehicle Type Classification Using PCA with Self-Clustering.” In: *2012 IEEE International Conference on Multimedia and Expo Workshops*. July 2012, pp. 384–389. DOI: 10.1109/ICMEW.2012.73.

[69] I Ketut Eddy Purnama Reza Fuad Rachmadi. “Vehicle Color Recognition using Convolutional Neural Network.” In: (2018). arXiv: 1510.07391 [cs.CV]. URL: https://arxiv.org/abs/1510.07391.

[70] Pan Chen, Xiang Bai, and Wenyu Liu. “Vehicle Color Recognition on an Urban Road by Feature Context.” In: *Intelligent Transportation Systems, IEEE Transactions on* 15 (Oct. 2014), pp. 2340–2346. DOI: 10.1109/TITS.2014.2308897.

[71] Nakhoon Baek et al. “Vehicle Color Classification Based on the Support Vector Machine Method.” In: vol. 2. Aug. 2007, pp. 1133–1139. ISBN: 978-3-540-74281-4. DOI: 10.1007/978-3-540-74282-1_127.

[72] Youngmin Kim, Donghwa Kang, and Hyeongboo Baek. “A 2-Stage Model for Vehicle Class and Orientation Detection with Photo-Realistic Image Generation.” In: June 2025. DOI: 10.48550/arXiv.2506.01338.

[73] Paul Rybski et al. “Visual classification of coarse vehicle orientation using Histogram of Oriented Gradients features.” In: July 2010, pp. 921–928. DOI: 10.1109/IVS.2010.5547996.

[74] Mehmet KUNDURACI and Humar Kahramanli Örnek. “Vehicle Brand Detection Using Deep Learning Algorithms.” In: *International Journal of Applied Mathematics Electronics and Computers* 7 (Sept. 2019), pp. 70–74. DOI: 10.18100/ijamec.578497.

[75] Xiaoli Jiang et al. “Vehicle Logo Detection Method Based on Improved YOLOv4.” In: *Electronics* 11.20 (2022), p. 3400. DOI: 10.3390/electronics11203400. URL: https://www.mdpi.com/2079-9292/11/20/3400.

[76] Jie Fang et al. “Fine-Grained Vehicle Model Recognition Using a Coarse-to-Fine Convolutional Neural Network Architecture.” In: *IEEE Transactions on Intelligent Transportation Systems* (Nov. 2016), pp. 1–11. DOI: 10.1109/TITS.2016.2620495.

[77] Linjie Yang et al. “A large-scale car dataset for fine-grained categorization and verification.” In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3973–3981. DOI: 10.1109/CVPR.2015.7299023.

[78] Dataset Ninja. *Visualization Tools for Vehicle Dataset for YOLO Dataset*. https://datasetninja.com/vehicle-dataset-for-yolo. Computer Vision Tools. visited on 2025-05-22. May 2025. URL: https://datasetninja.com/vehicle-dataset-for-yolo.

[79] RoboFlow and Bob Enwmj. *Plate Detector (ALKO1) Dataset*. RoboFlow Universe. Accessed: 2025-06-10. 2024. URL: https://universe.roboflow.com/bob-enwmj/plate-detector-alko1/dataset/1.

[80] Yarab and RoboFlow. *Cars Logo Dataset (Version 7)*. RoboFlow Universe. Accessed: 2025-06-10. 2023. URL: https://universe.roboflow.com/yarab/cars-logo-my5mz/dataset/7.

[81] JP0909. *Car Logos Dataset*. Kaggle Dataset. Accessed: 2025-06-10. 2020. URL: https://www.kaggle.com/datasets/jp0909/car-logos/.

[82] Riotulab. *Car Make, Model and Generation Dataset*. Kaggle Dataset. Accessed: 2025-06-10. 2023. URL: https://www.kaggle.com/datasets/riotulab/car-make-model-and-generation/data.

[83] Landry KEZEBOU. *VCOR (Vehicle Color Recognition) Dataset*. Kaggle. Accessed: 2025-05-19. 2020. URL: https://www.kaggle.com/datasets/landrykezebou/vcor-vehicle-color-recognition-dataset.

[84] Wong. *Front and Rear of Car Dataset*. https://universe.roboflow.com/wong-l7j5n/front-and-rear-of-car. Accessed: 2025-05-19. 2023.

[85] Sohrab Namazi Nia and Frank Y Shih. "Medical X-Ray Image Enhancement Using Global Contrast-Limited Adaptive Histogram Equalization." In: (2024). DOI: 10.1142/s0218001424570106.

[86] Richard S. Hunter. "Photoelectric Color-Difference Meter." In: *Journal of the Optical Society of America* 38.7 (), p. 661.

[87] Karel Zuiderveld. "Contrast Limited Adaptive Histogram Equalization." In: *Graphics Gems IV*. Ed. by Paul Heckbert. Academic Press, pp. 474–485. ISBN: 0-12-336155-9.

[88] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. 4th. Pearson, 2018.

[89] Hanae Moussaoui et al. "Enhancing automated vehicle identification by integrating YOLO v8 and OCR techniques for high-precision license plate detection and recognition." In: *Scientific Reports* (2024).

[90] Guangzhen Yao et al. "HP-YOLOv8: High-Precision Small Object Detection Algorithm for Remote Sensing Images." In: *Sensors* 24.15 (2024), p. 4859. DOI: 10.3390/s24154858. URL: https://www.mdpi.com/1424-8220/24/15/4858.

[91] Jane Torres. *How to modify yolov8 architecture?* Accessed: May 19, 2025. 2024. URL: https://yolov8.org/how-to-modify-yolov8-architecture/.

[92] Abdussalam Elhanashi et al. "TeleStroke: real-time stroke detection with federated learning and YOLOv8 on edge devices." In: *Journal of Real-Time Image Processing* 21 (2024), p. 121. DOI: 10.1007/s11554-024-01500-1. URL: https://doi.org/10.1007/s11554-024-01500-1.

[93] Gonzalez et al. *Digital Image Processing*. Prentice Hall.

[94] Minghao Li et al. "TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models." In: *arXiv preprint arXiv:2109.10282* (2021).

[95] Keras Team. *Keras: The Python Deep Learning API*. https://keras.io/. Accessed: 2025-05-21. 2024.

[96] SQLite. *SQLite Home Page*. Official website of SQLite, a self-contained, serverless, zero-configuration SQL database engine. 2023. URL: https://www.sqlite.org/ (visited on 11/15/2023).

[97] Gunjan Raj et al. "Python current trend applications: an overview." In: *ResearchGate* (2020). URL: https://www.researchgate.net/publication/344569950_Python_current_trend_applications-an_overview.

[98] Google. *Foire aux questions - Colaboratory*. Accessed: 2025-05-21. 2023. URL: http://research.google.com/colaboratory/faq.html?hl=fr.

[99] Jupyter Project. *Jupyter Documentation*. https://jupyter.org/. Accessed: June, 2025. 2023.

[100] TensorFlow Team. *TensorFlow Tutorials*. https://www.tensorflow.org/tutorials?hl=fr. Accessed: 2025-05-21. 2024.

[101] PyTorch Team. *PyTorch: An Open Source Machine Learning Framework*. Accessed: 2025-05-21. 2025. URL: https://pytorch.org/projects/pytorch/.

[102] Qingyun Ma. "Research on recognizing required items based on OpenCV and machine learning." In: *Proceedings of the 2022 International Conference on Information Technology in Education and Management Engineering (ITEME2022)*. Vol. 140. SHS Web of Conferences. Accessed: 2025-05-21. EDP Sciences, 2022, p. 01016. DOI: 10.1051/shsconf/202214001016. URL: https://www.shs-conferences.org/articles/shsconf/pdf/2022/10/shsconf_iteme2022_01016.pdf.

[103] Sandro Tosi. *Matplotlib for Python Developers*. Accessed: 2025-05-21. Packt Publishing, 2009. URL: https://theswissbay.ch/pdf/Gentoomen%20Library/Programming/Python/Matplotlib%20for%20Python%20Developers%20%282009%29.pdf.

[104] NumPy Developers. *What is NumPy?* Accessed: 2025-05-21. 2024. URL: https://numpy.org/doc/stable/user/whatisnumpy.html.

[105] MYSCALE Team. "Streamlit vs Gradio: The Ultimate Showdown for Python Dashboards." In: *Python Visualization Conference*. Online; Accessed September 2024. PYVIS, Sept. 2024, pp. 1–15. URL: https://myscale.com/python-dashboards.

[106] Real Python. *Image Processing with the Python Pillow Library*. https://realpython.com/image-processing-with-the-python-pillow-library/. Accessed: 2025-05-21. 2024.

[107] Abdussalam Elhanashi et al. "TeleStroke: real-time stroke detection with federated learning and YOLOv8 on edge devices." In: *Journal of Real-Time Image Processing* 21 (2024), p. 121. DOI: 10.1007/s11554-024-01500-1. URL: https://doi.org/10.1007/s11554-024-01500-1.

A Project Presentation

A.1 The Project Idea (Proposed Solution)

The growing need for advanced, secure, and automated vehicle access control systems is especially evident in Algeria, where intelligent solutions are still not widely used. In contrast to many developed countries, Algeria lacks widespread implementation of automated systems that integrate artificial intelligence for vehicle identification. Most access control systems in the country, whether at universities, residential compounds, government institutions, or corporate sites, are still managed manually, relying on security personnel without technological support. Even the few developed solutions available focus solely on basic license plate recognition, failing to address full vehicle identification, such as brand, model, orientation, and color, which are essential for enhanced security and fraud prevention.

This project directly responds to that gap by proposing a comprehensive AI-based vehicle recognition system specified to Algeria's specific context. The system goes beyond license plate recognition, offering complete vehicle identification using advanced deep learning models. It integrates deep learning models for vehicle detection, plate localization, orientation estimation, logo recognition, models for brand, model, and color classification. To ensure strong performance, we collected and utilized multiple datasets, including those featuring Algerian vehicles and license plate formats. This integrated approach allows the system to operate effectively under real-world conditions, achieving high accuracy and robustness.

The project aims to modernize access control practices in Algeria, supporting smart infrastructure initiatives and laying the foundation for intelligent, automated vehicle management in both public and private sectors.

A.2 The Proposed Values

- **Security:** Only authorized vehicles gain access based on intelligent, AI-powered verification, replacing traditional methods (e.g., guards, badge readers).
- **Automation:** Reduces manual labor through real-time detection, entry verification, and logging, enabling efficient tracking of vehicles in sensitive areas.
- **Speed:** Instant recognition and decision-making ensure smoother traffic flow and reduced wait times at entry points.
- **Scalability:** Easily adaptable to various environments including residential zones, industrial sites, educational campuses, and public institutions.
- **Modernization:** Supports Algeria's smart city goals by introducing intelligent access control solutions that align with global innovation trends.
- **Localization:** Fully supports Algerian license plate formats and regional vehicle features for improved accuracy and relevance.
- **Analytics:** Provides valuable insights for administrators, such as frequent visitors, entry times, and suspicious activity alerts.

- **Affordability:** Offers an efficient and cost-effective solution for small and medium-sized enterprises (SMEs) seeking modern access control.
- **Performance:** Ensures reliable and accurate recognition results by leveraging deep learning models optimized for high precision.
- **Cost Reduction:** Reduces operational costs by minimizing human intervention and automating key access control tasks.
- **Design:** Features an intuitive and user-friendly interface tailored to administrators and security personnel, ensuring ease of use and effective monitoring.

A.3 The working Team

- **Ikram Merour:** Specializes in computer vision, deep learning, artificial intelligence, and interface development. Her role focused on system architecture, YOLO/TrOCR implementation, and deep learning training.
- **Balkiss Boulahia:** Specializes in computer vision, deep learning, artificial intelligence, and interface development. Her role focused on database management, system architecture, and the deployment pipeline.
- **Dr. Hassina Bouressace:** Supervisor and consultant, an expert in the field of computer science, responsible for project management. She provides academic and technical guidance, oversees the realization of the project, and offers strategic advice.

A.4 The Project's Objectives

1. **Short-Term (Year 1):**
    - Launch the pilot version of the system in small to medium entry points, such as residential areas, office buildings, and local parking lots.
    - Collect performance data and user feedback to assess functionality and system stability.
    - Aim for successful integration and acceptance within limited, controlled environments.
2. **Medium-Term (Years 2-3)**
    - Expand the deployment to include corporate university campuses, hotel entry points, and early-stage smart city projects in Algeria.
    - Establish partnerships with local infrastructure providers and public institutions.
    - Enhance system features based on initial deployment feedback, focusing on scalability and integration.
3. **Long-Term (Year 4+):**
    - Scale the system at a national level, integrating it with Algeria's transportation networks, law enforcement databases, and toll systems.

- Enable smart traffic management, automatic violation detection, and public safety monitoring.
- Position the system as a core component of Algeria's intelligent infrastructure and smart city vision.

A.5 Project Completion Schedule

Table 9: Project tasks timeline: (Mo) Month

Task	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**
Research Study & Planning				*	*	
Dataset Preparation & Training	*	*	*			
System Development			*	*	*	*
Testing & Optimization				*	*	*
Marketing & Deployment					*	*

B Innovative Aspects

B.1 First of Its Kind in the Region:

- **First of Its Kind in the Region:** The initial framework in Algeria integrates license plate recognition, logo & model identification, color identification, and vehicle orientation assessment for an all-encompassing, automated access management.
- A cohesive system tailored for high-security environments, including secure parking areas, educational institutions, and industrial zones.

B.2 Integration with Other Smart Systems:

- **Interoperability** :Can be incorporated into an international security management framework (such as video monitoring, alarm systems, and identity verification).
- Connection to governmental or corporate databases for automatic verification of whether a vehicle is reported stolen or unauthorized.

B.3 Dynamic Access Policies:

- **Dynamic Access Rules:** Capability to establish access schedules, prohibited zones, or particular guidelines based on the color, model or type of vehicle.
- **Temporary or Visitor Access:** Automatic creation of QR codes or temporary passes associated with an identified vehicle.

B.4 Eco-Friendly Deployment:

- **Energy-conserving equipment:** Utilization of low-energy cameras and environmentally sustainable mini-PCs.

- **Reduction of paperwork:** Achieved by the total digitization of access and vehicle records.

B.5 Advanced User Interface & Reporting:

- **Interactive Gradio Interface or WebApp:** Presents identified vehicle information (image, timestamp, registration number, manufacturer, type) along with options for PDF/Excel export.

C Strategic Market Analysis

C.1 Market Segment

- Governmental and private institutions with secured zones.
- Residential buildings, corporate offices, hospitals, and university campuses.

C.2 Target Clients:

- Entry point of the university or residence.
- Parking management companies, smart city (Residential gated communities).
- Property managers of gated communities and apartment complexes.
- Law enforcement agencies (traffic, border control).

C.3 Competitive Intensity

- **Direct Competitors:** Basic license plate recognition system providers.

Our Strengths:

- Broader functionality beyond plate detection.
- High adaptability to custom needs.
- Cloud or local deployment options.

C.4 Marketing Strategy

- **Academic Collaborations:** Partner with universities and engineering schools to promote the system as a research and innovation tools, increasing credibility and adoption in public institutions.

- **Pilot Programs and Real-World Demonstrations:** Launch live pilots in strategic locations (corporate parking, universities, gated communities) to showcase the system in action. Provide real data on reduced waiting time, increased security, and automated reporting.

- **Strategic Partnerships:** Collaborate with parking solution providers, smart city developers, and security agencies to integrate our system as a core module of broader infrastructure solutions.

- **Smart Integration with Existing Equipment:** Promote the system's ability to plug existing camera infrastructure, minimizing client investment and accelerating the deployment process.

D Production and Organization Plan

D.1 Production Steps

System Development:

- Dataset collection and training (vehicle, plates, logos, color, models)
- Interface development (dashboard, input/output stream,)
- Design Gradio-based dashboard.
- Deployment and support.

D.2 Supply

Hardware: Cameras, servers.
Software Tools: PyTorch, YOLOv8, Gradio, SQLite/PostgreSQL.

D.3 Partnerships

- Collaboration with universities and AI research labs for models improvement
- With national security agencies. private parking providers, local tech firms

D.4 Jobs Created

- AI engineers, full-stuck developers, data annotators.
- field technicians, sales/support teams.

E Financial Plan

E.1 Costs and Charges

Because there is a variety of unique perspectives that need to be organized and analyzed, it is essential to recognize all essential costs and investments when engaging with clients. This includes the annual, fixed, and supplementary expenses alongside the basic costs. The key elements to consider are as follows:

E.2 Initial Costs

Infrastructure:

- Establishing a rented area specifically for software creation, system management, and operational oversight of the intelligent vehicle access management platform.
- Introducing both physical and digital security protocols, which involve setting up servers, workstations, surveillance cameras, and high-speed network connections to guarantee dependable and instantaneous monitoring.
- Making monthly payments for cloud services to host a secure database containing authorized vehicle information, surveillance videos, and access logs.

Equipment:

- **Computers and Workstations:** Acquisition of modular workstations intended for developers, security personnel, data processing specialists, and administrative personnel tasked with overseeing the platform.
- **Servers and Cloud Platforms:** Versatile cloud solutions that can handle incoming video feeds, retain license plate recognition information, and oversee access credentials management.
- **Networking Devices:** Routers, switches, IP cameras, access control units, and other vital networking apparatuses to guarantee uninterrupted communication among all components of the system.
- **Software Tools:** Licenses for development environments such as Python, TensorFlow, YOLO, TrOCR, data visualization applications, and database management platforms.
- **Security and Backup Systems:** Backup hardware solutions designed for automated data preservation, ensuring safeguards against cyber threats and unintentional data loss.
- **Office supplies:** Procure essential consumables that encompass office furniture, document storage options, printing resources, and additional supplies crucial for operational teams.

Technology:

- **Intelligent Detection Models:** Implementation of sophisticated models like YOLOv8 for identifying license plates, logos, colors, and vehicle orientations, paired with TrOCR for superior optical character recognition performance.
- **Data Management Systems:** Robust and scalable databases (e.g., MySQL, MongoDB, sqLite) utilized for archiving access logs, vehicle images, and results from recognition processes.
- **Integration Tools:** API solutions and integration instruments that link the intelligent access system with physical security elements such as automatic barriers, RFID sensors, or alarm systems.

- **User Interfaces:** Sleek and responsive web and mobile platforms crafted to showcase detection outcomes, register authorized vehicles, and examine access histories in an intuitive manner.
- **Analytics and Reporting Tools:** ns, system engagement metrics, and incident notifications to support decision-makers in management operations.

E.3 Operational Costs

Personnel:

- **Salaries:** Compensation for personnel engaged in platform development, hardware setup, cloud service management, customer assistance, and operational oversight.
- **Continuous Training:** Regular training initiatives for team members to remain informed about the most recent progress in artificial intelligence, visual recognition, data protection, and smart security systems.

Logistics:

- **Cloud Hosting Services:** Ongoing payments for cloud services related to data storage, model inference processing, bandwidth usage, and the maintenance of platform availability.
- **Software Subscriptions:** Recurring monthly or yearly charges for vital software licenses, analytical platforms, and third-party services connected to the system.

Marketing and Customer Service:

- **Marketing Compaigns:** Funding for online marketing initiatives, content development, focused advertisements, and public relations to raise awareness of the access management platform among prospective clients (such as parking facilities, private businesses, and government establishments).
- **Customer Service Operations:** Expenses associated with handling user inquiries, addressing technical questions, solving access problems, and guaranteeing a positive user experience.

E.4 Other Costs

- Insurance coverage for liabilities arising from incidents related to access control failures or breaches of data.
- Coverage for data loss and implementation of backup systems to guarantee resilience and operational continuity.
- Legal costs incurred for obtaining essential licenses and regulatory permits.
- Expenses associated with adhering to national data privacy regulations and digital security laws.

E.5 Recurring Costs

- Consistent upkeep of technical apparatus, monitoring setups, and management systems.
- Renewals of software licenses, regulatory permits, and cloud service subscriptions.
- Scheduled enhancements to system software and hardware to adapt to technological advancements and changing security risks.

E.6 Methods and Sources of Obtaining Financing

A) Internal Financing:

- Reinvesting earnings from current operations inside the company.
- Setting aside company funds exclusively for advancements in smart infrastructure.

B) External Financing:
Bank Loans:

- Extended loans for substantial upfront expenditures.
- Credit lines for operational and fluctuating expenses.

Investors:

- Looking for investors who are keen on innovative AI technologies within security and intelligent city frameworks.
- Forming alliances with venture capitalists who concentrate on artificial intelligence, computer vision, and advanced infrastructure.
- Investigating opportunities for business growth to broaden the platform's reach in various markets.

Subsidies and Aid:

- Seeking government grants and innovation subsidies aimed at the advancement of smart city initiatives and AI-driven technologies.
- Engaging with technology accelerators and incubators that provide mentorship, exposure, and capital to emerging startups in the fields of security and automation.

Crowdfunding:

- Initiating a crowdfunding effort to engage small investors and supporters who appreciate advancements in digital technology and urban security.

E.7 Obtaining Reimbursement

Scheduling Repayments:

- A repayment plan should comprise all outstanding amounts with specific due dates, grace periods, and applicable interest rates for each type of financing method.

Cash Flow Management:

- Anticipate cash flow to guarantee the ability to repay.
- Modify spending and revenue forecasts to prevent issues.

Payment Table:

- Detailed cash flow projections will be employed to ensure adequate liquidity and prevent late payments or financial constraints.
- Adjustments to expenses and income will be made as necessary to sustain financial balance.

Table 10: Subscription data: one year (N)

Year	Client Sites	Site Subscription (DA/year)	SMEs Users	User Access Fee (DA/year)	Subscription Revenue (DA)	User Revenue (DA)	Total Revenue (DA)
N	20	10,000	200	4,500	200,000	900,000	1,100,000
N+1	40	11,000	400	4,750	440,000	1,900,000	2,340,000
N+2	80	12,000	600	5,000	960,000	3,000,000	3,960,000
N+3	120	13,000	1,000	5,250	1,560,000	5,250,000	6,810,000
N+4	150	14,000	1,500	5,500	2,100,000	8,250,000	10,350,000

F Steps to obtain the service

our system has four different interfaces:

1. The welcome page
2. The main interface (Vehicle Detection)
3. Access control management interface
4. finally database interface

with a special focus on the main modules. The figure below shows the welcome page of our application.

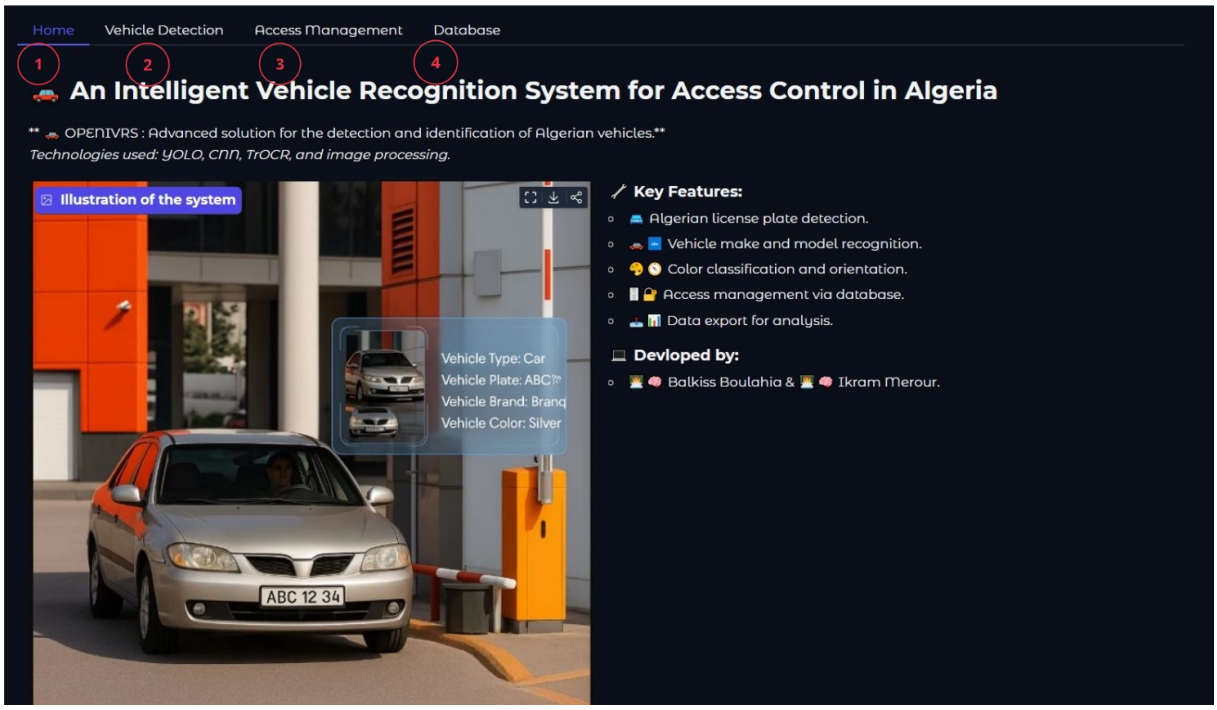


Figure 29: The home interface of our system.

After clicking on the "Vehicle Detection", the basic interface shown in the image below appears. (See Figure 30)

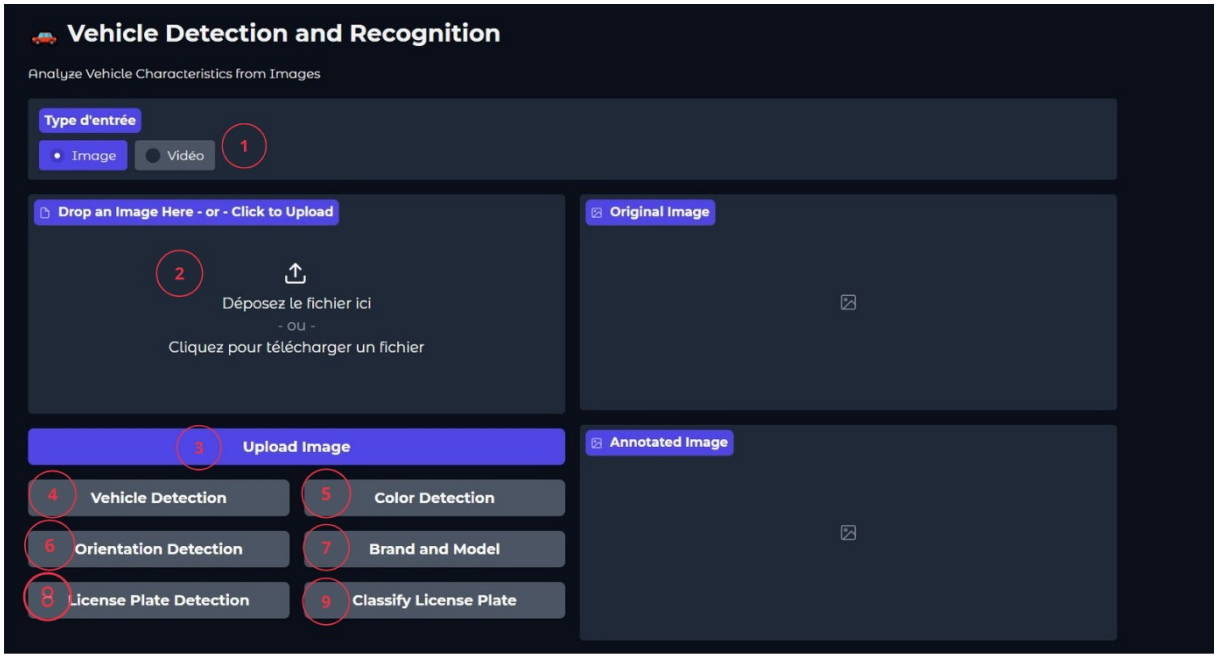


Figure 30: Basic interface of our system

The main modules of our application are summarized by the following numbers:

1. The initial radio button allows the user to select the type of input (image or video).
2. The second button enables the user to upload the file (image or video).

3. The third button is designed to perform the task of vehicle detection and type recognition and subsequently draw a bounding box around the identified vehicle.

4. The fourth button is designed to perform the task of color recognition.

5. The fifth button is designed to perform the task of vehicle orientation detection and recognition and subsequently draw a bounding box around the identified vehicle part.

6. The Sixth button is designed to perform the task of logo detection, recognition and model recognition subsequently draw a bounding box around the identified logo.

7. The Seventh button is designed to perform the task of license plate detection and recognition and subsequently draw a bounding box around the identified license plate and also on its characters.

8. The Eighth button is responsible for classify the recognized plate.

After clicking on the "Access Management", the interface shown in the image below appears.
(See Figure 31)

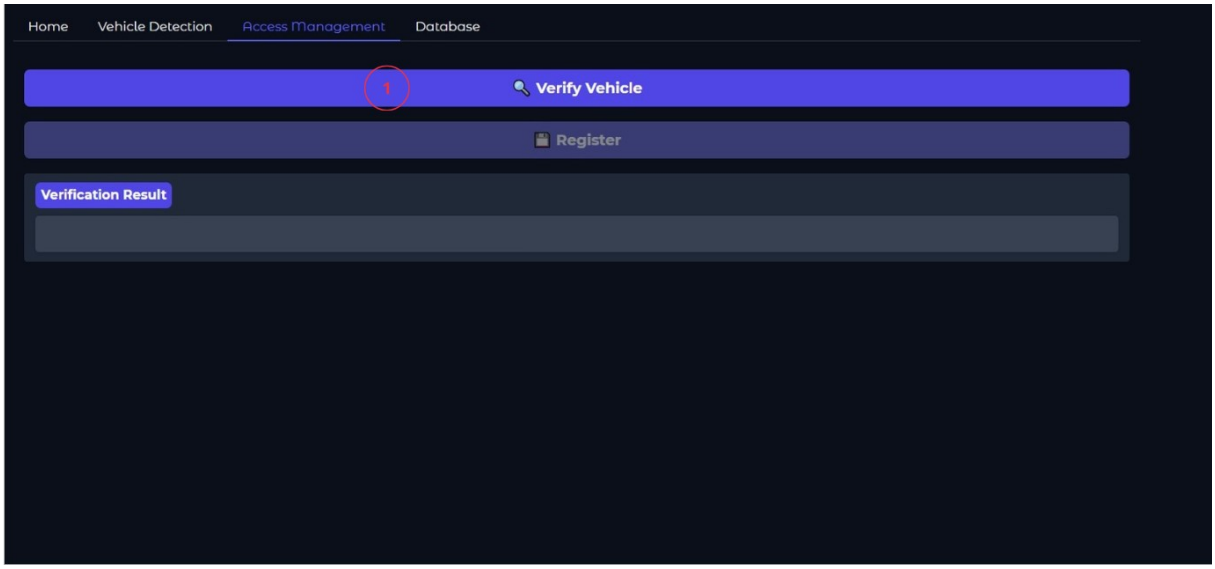


Figure 31: Access control interface of our system

(Figure 32) represents the case when the vehicle is not existed in the database.

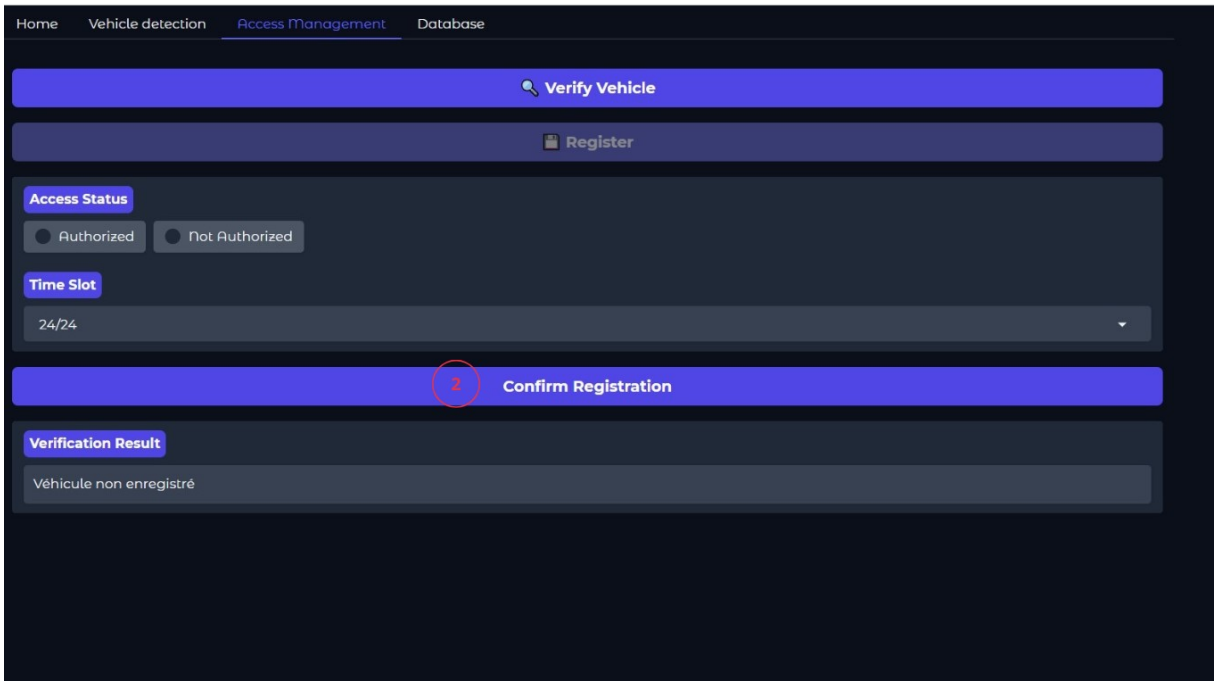


Figure 32: Access control interface of our system(new vehicle)

The main modules of this part are summarized by the following numbers:

1. first we click on the button verify vehicle, to check if it already exist or not.
2. second button confirme enables the user to save the information of new vehicles.

After clicking on the ”database”, the interface shown in the image below appears. (See Figure 33)

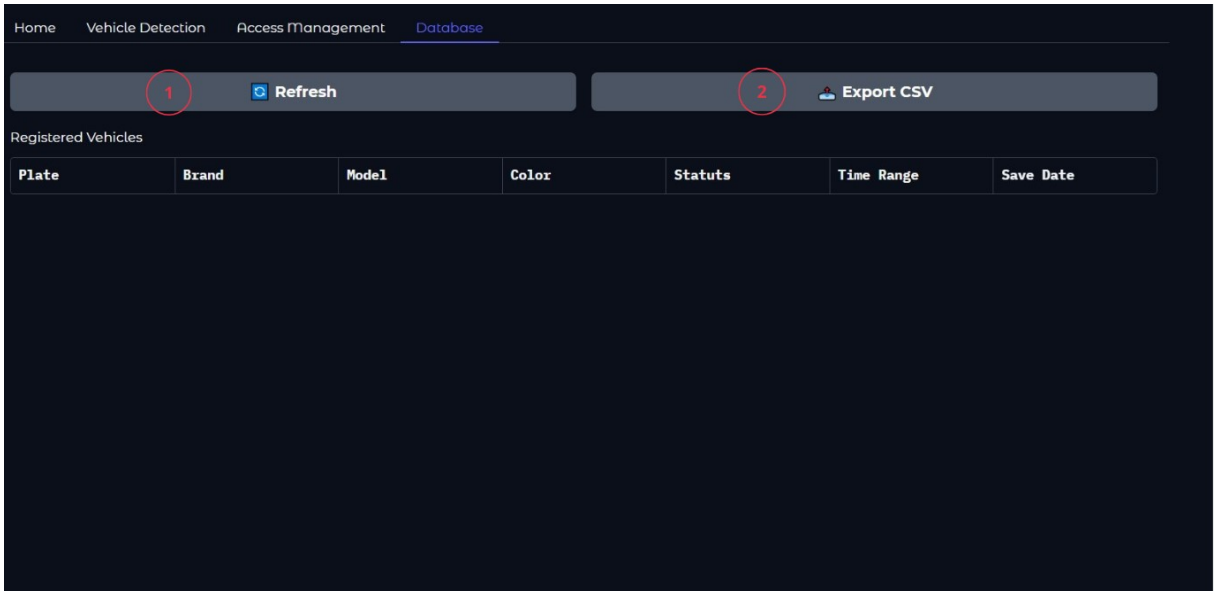


Figure 33: Database interface of our system

The main modules of this part are summarized by the following numbers:

1. the button ”refresh” for refresh the database tables.
2. the button ”export CSV” if we want to upload another database file.

G Usage scenario

In this section, we will provide an overview of the working principles of our system. We will describe the different stages involved in vehicle recognition for control access system:

1. To commence the vehicle recognition process, we start by opening a (image/video) file using the command "Upload file" This allows us to access the desired images and videos and utilize it for further analysis and detection procedures. As shown in Figure 34

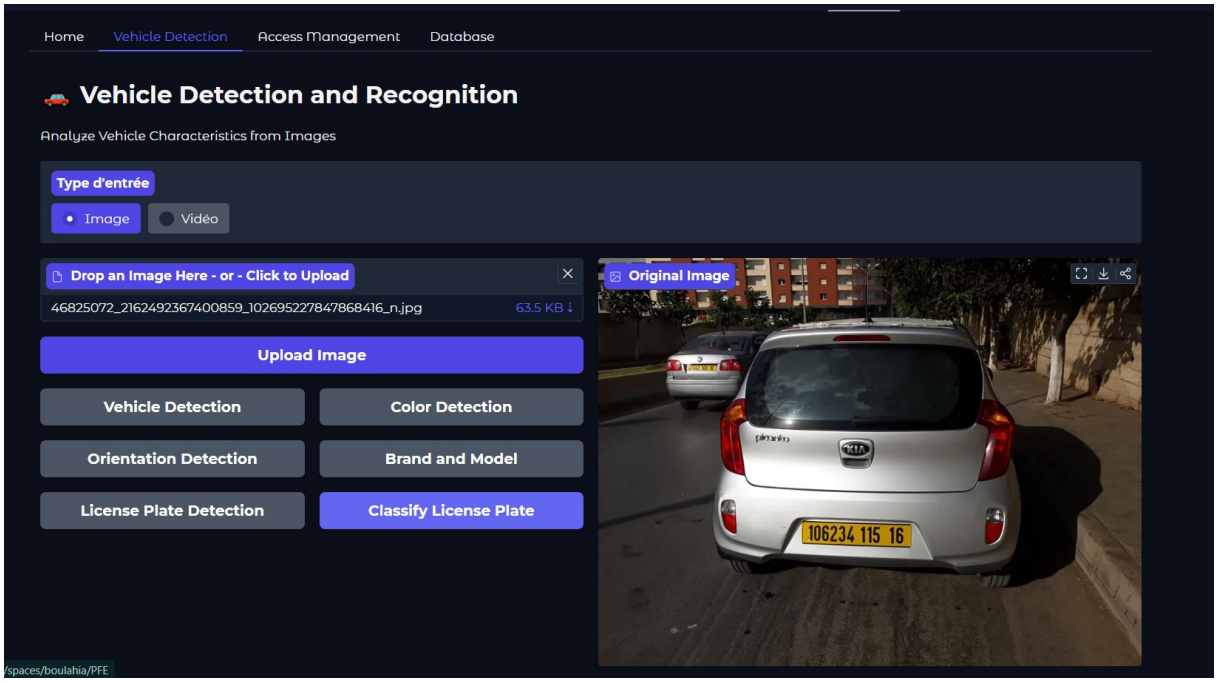


Figure 34: Input selection (image or video)

If the input type is a video we should extract the frames before we start the vehicle identification process, as shown in Figure 35

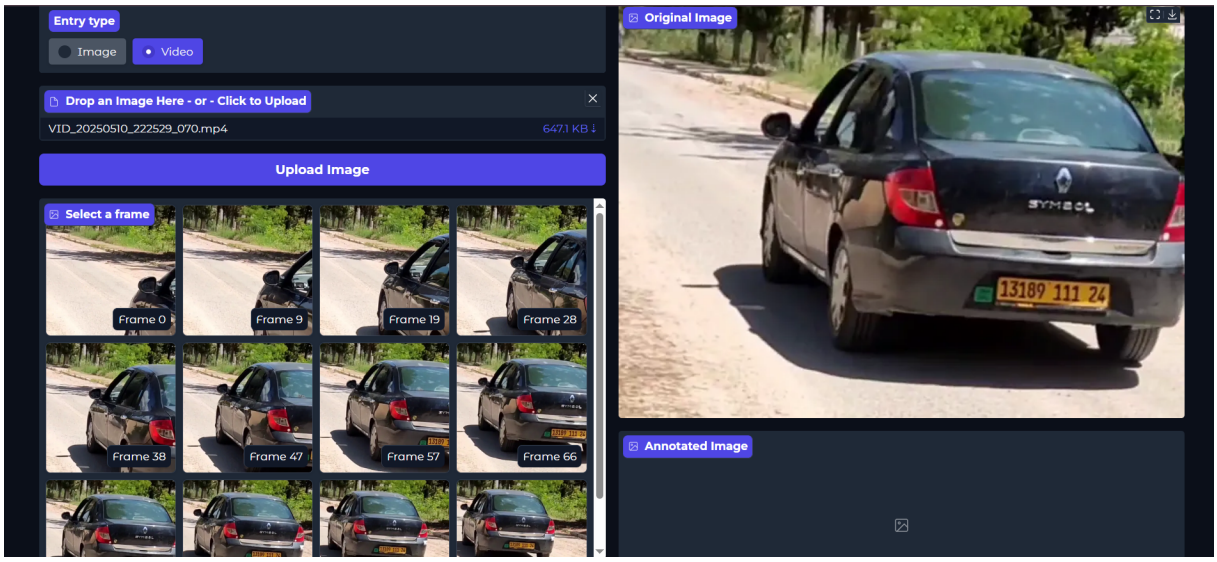


Figure 35: Frames extraction from input video

2. after opening the file successfully. When we click the "vehicle detection" button, the process of identifying the car in the opening picture or video begins, the YOLOv8 model which has already been trained on a dataset allows for precise and effective vehicle detection. The outcome displayed in Figure 36

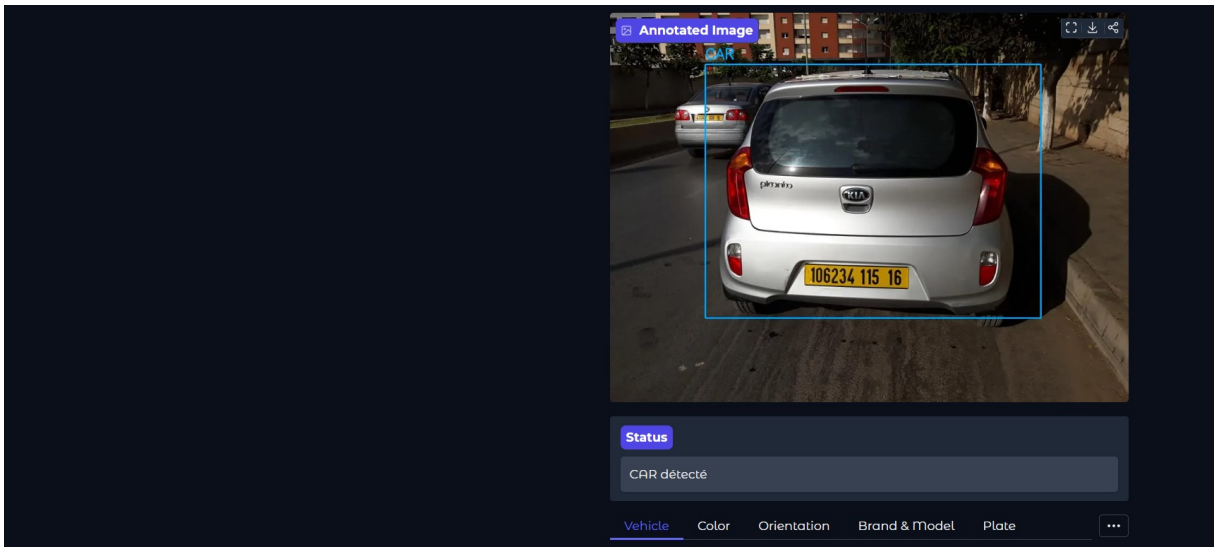


Figure 36: Vehicle detection & Recognition

3. When we click the "Color detection" button, the process of prediction the car color begins, the CNN model which has already been trained on a dataset allows for precise and effective color prediction. The outcome displayed in Figure 37

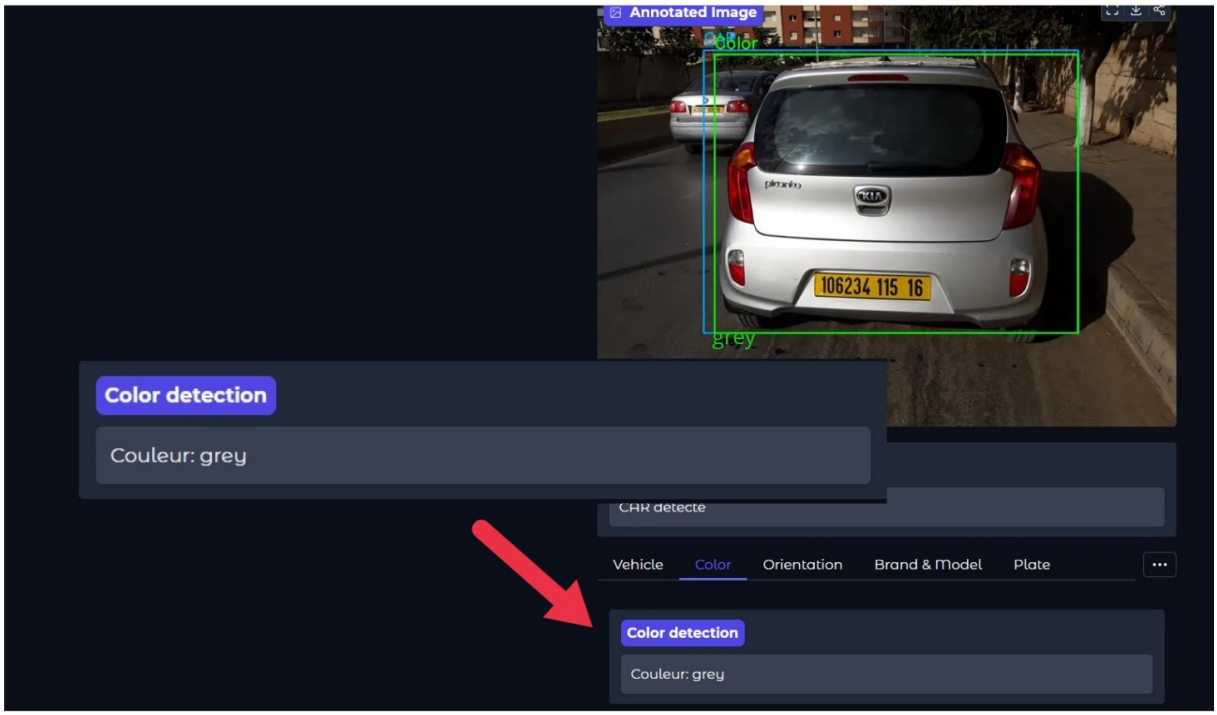


Figure 37: Color prediction

4. The button "Direction detection" start the process of identifying the vehicle orientation, the YOLOv8 model which has already been trained on a dataset allows for precise and effective vehicle detection. The outcome displayed in Figure 38

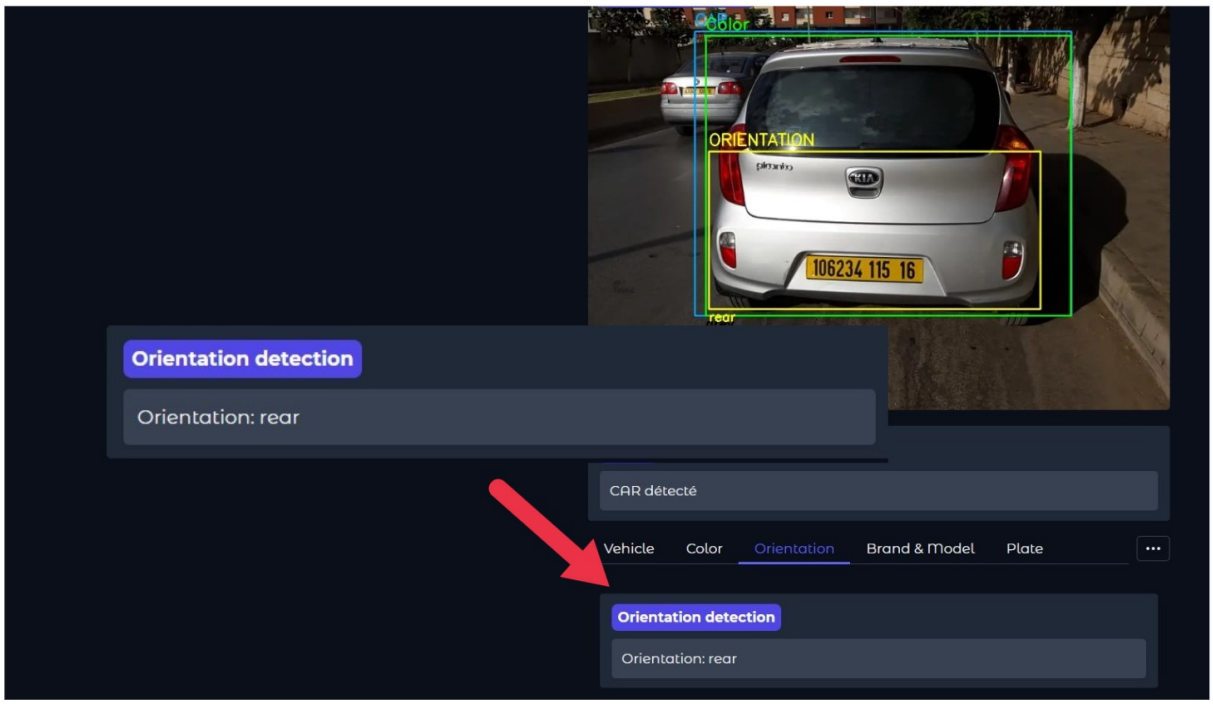


Figure 38: Orientation detection & recognition

5. When we click the "Brand and model" button, the process of prediction the brand and model begins, the CNN model which has already been trained on a dataset

allows for precise and effective prediction. The outcome displayed in Figure 52

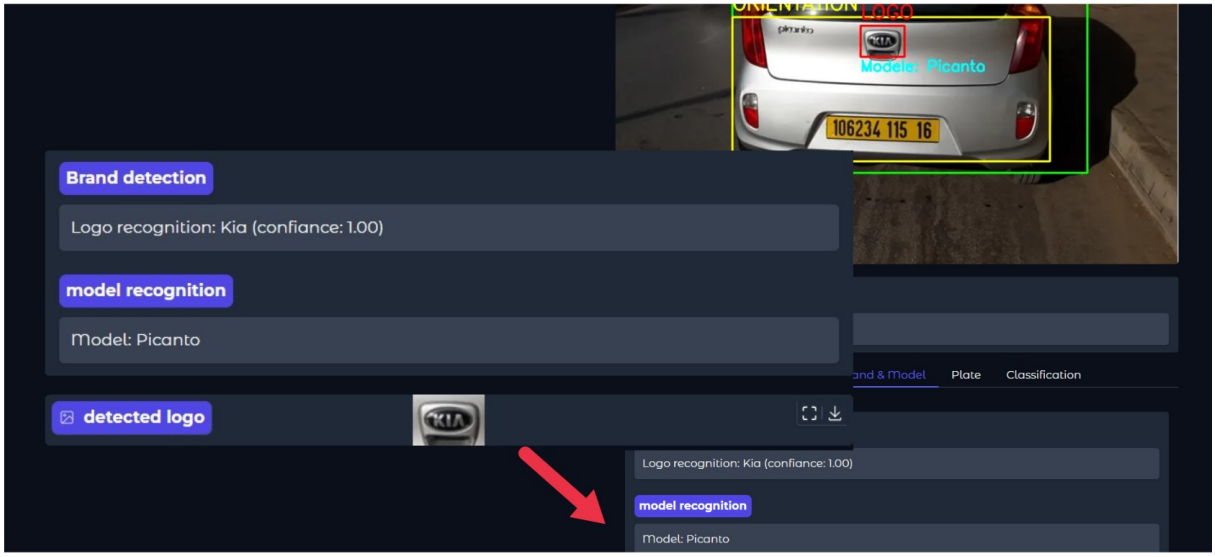


Figure 39: Make & Model Recognition

6. By selecting the ”plate detection” button, we initiate the process of detecting the license plate within the opened image and also recognize its number with TrOCR than show the list of the recognized characters. The result shown in Figure 40

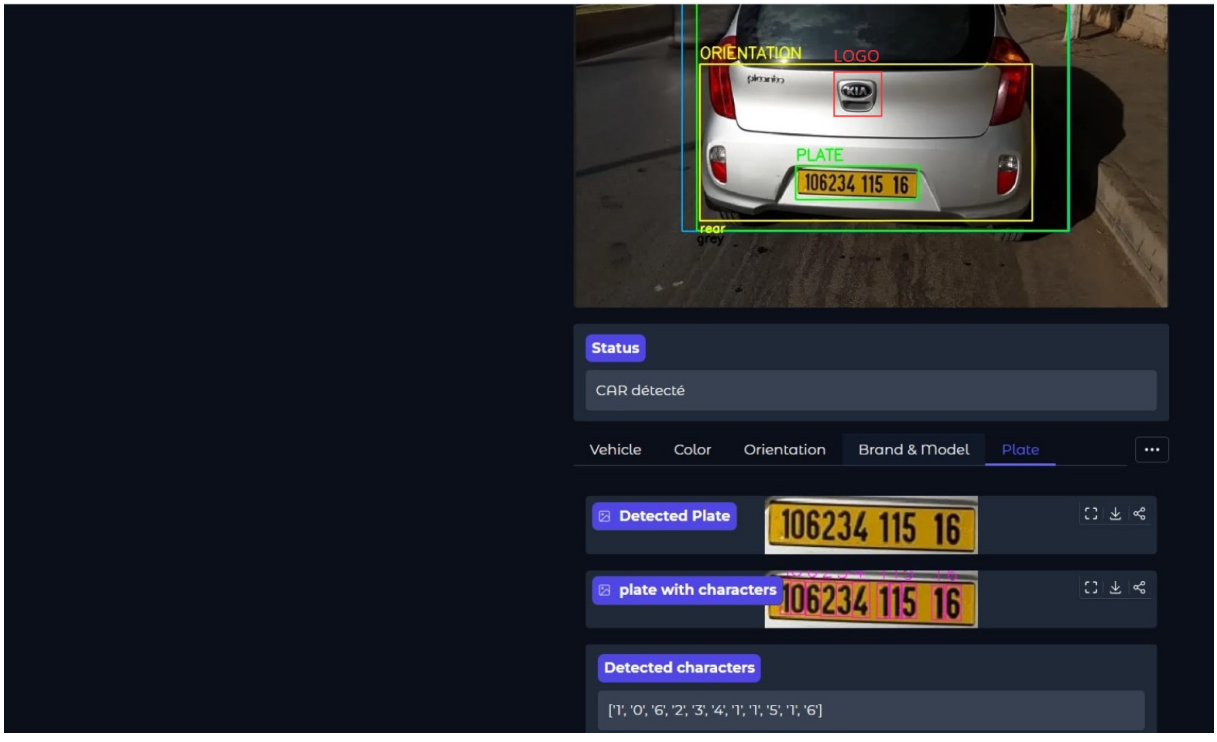


Figure 40: License plate detection and character recognition

7. A license plate classification procedure is carried out via the ”classify plate” button. It classifies the final two digits, the year of the license plate, and the type of vehicle after reading license plate numbers from the list of characters in each step. As

seen in Figure 41, the extracted data is subsequently shown, offering information about the license plate such as its translation, year, kind, and remaining characters. Additionally, Figure 42 shows an illustration of a non-Algerian plate.

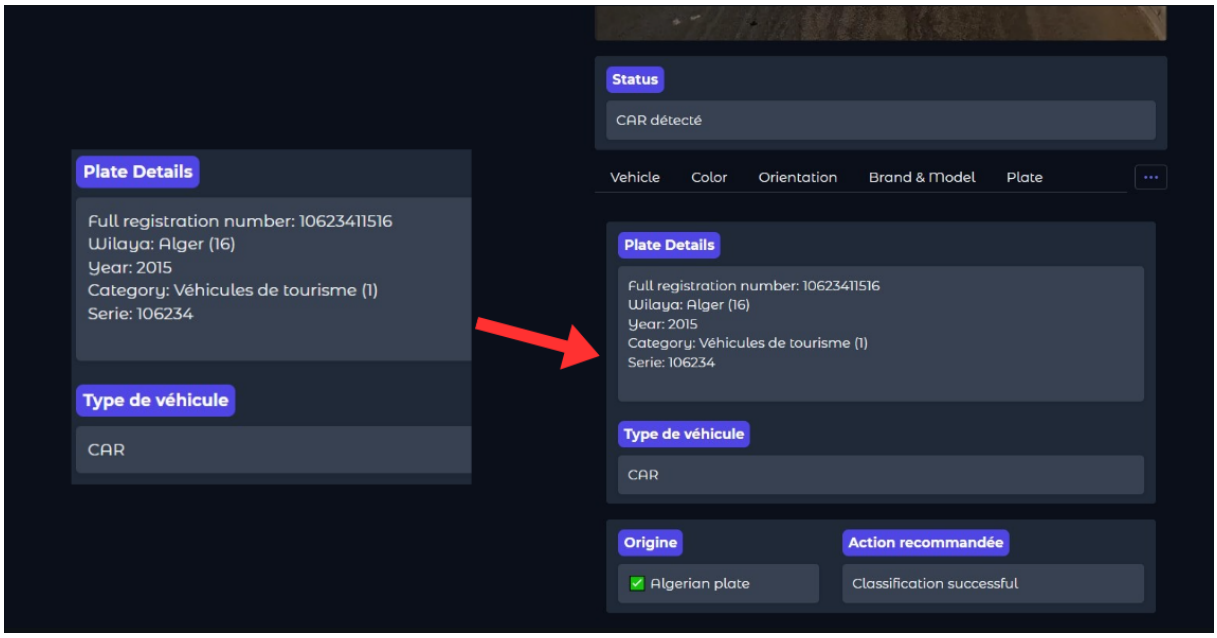


Figure 41: Character Classification.

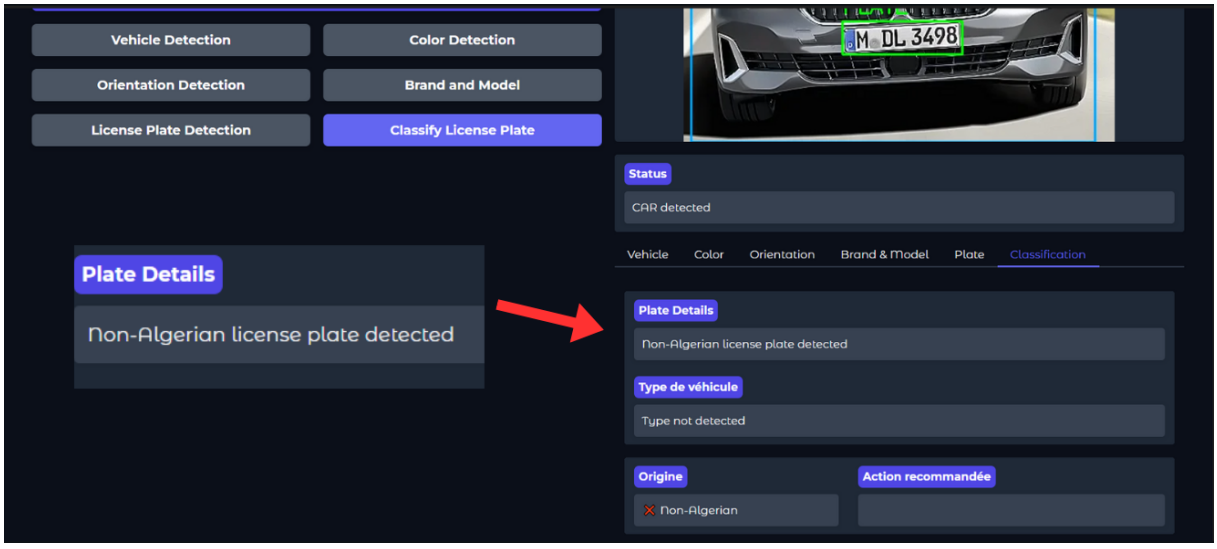


Figure 42: Example of non-Algerian plate.

8. The button ”verify vehicle” start the process of checking if the vehicle already exist or not. The outcome displayed in Figure 52. and after how to save it

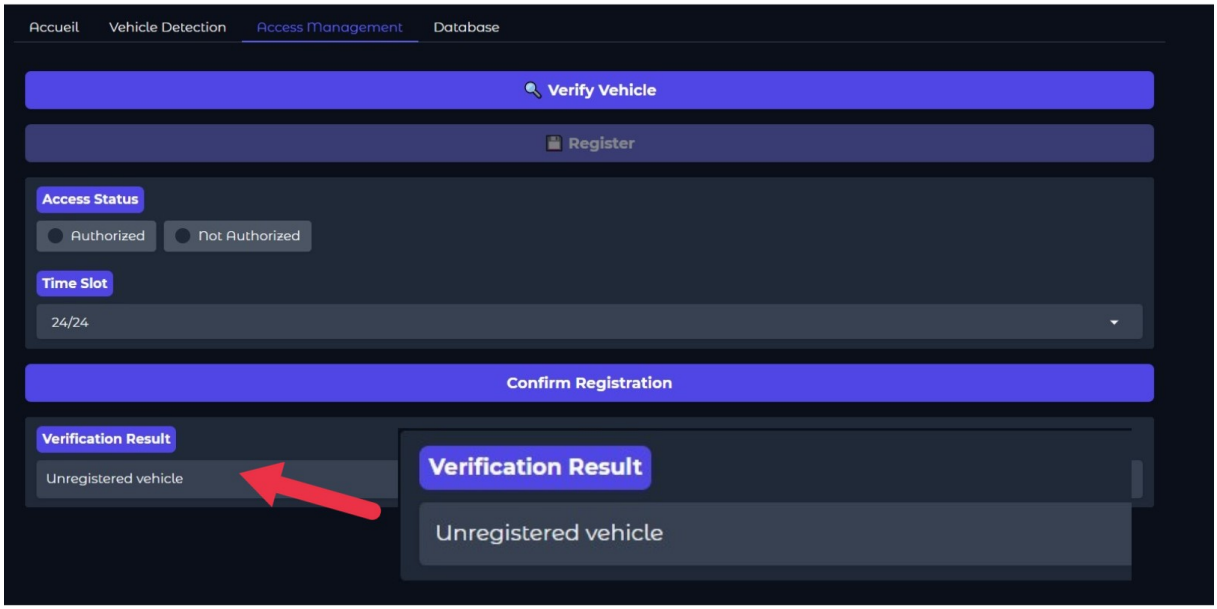


Figure 43: Vehicle existence verification

after that we select the Access status and the time slot and save it.

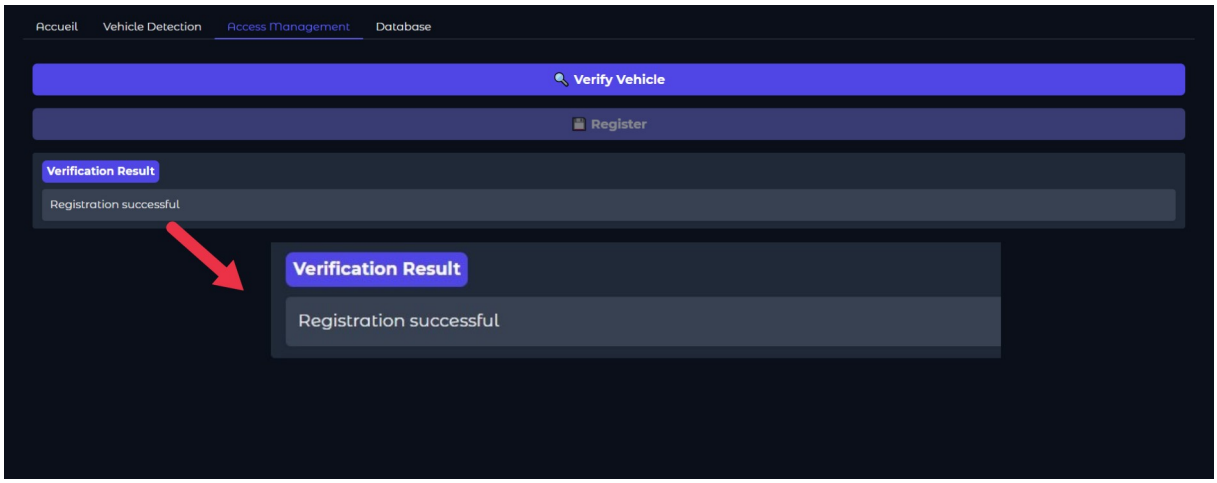


Figure 44: Save new vehicle entry

9. The button ”refresh” start the process of showing the list of vehicles and their information. The outcome displayed in Figure 45.

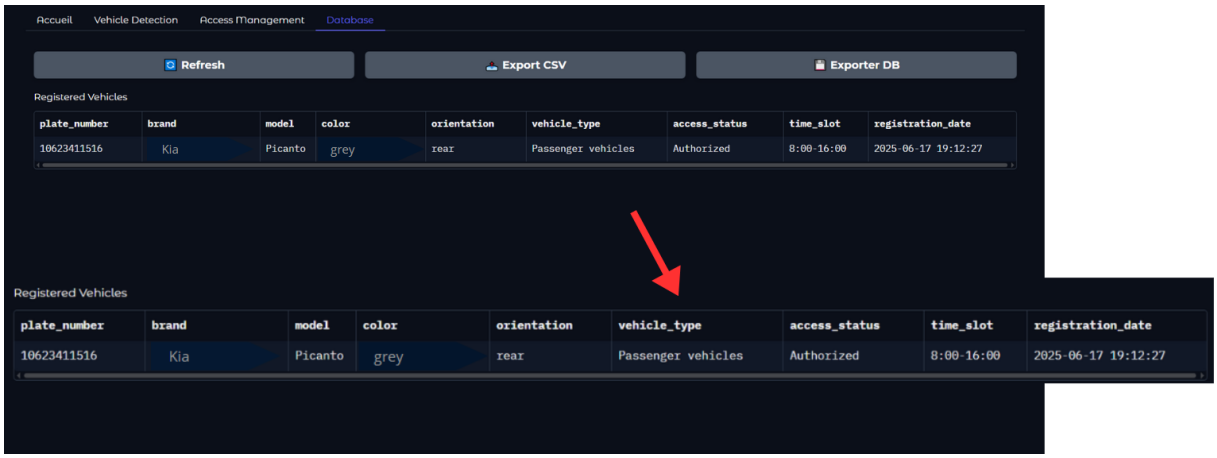


Figure 45: Vehicle list display

H Market Study and Validation

As part of the development of our smart solution for vehicle access management and control, a comprehensive market study was conducted to validate the project's relevance and potential adoption. The study was based on a structured questionnaire distributed to key stakeholders across multiple locations including:[**Groupement d'enterprise Hallici Mohammad amine et Hennad Hychem**], [**Genie civil et batiment/filiale sonatrach**], **and** [**SARL شركة ذات مسؤولية محدودة**].

The survey focused on three main aspects:

1. Whether the respondents currently had similar systems in place.
2. Their level of interest in specific features like automated license plate recognition, logo and model recognition, color and orientation prediction.
3. Their willingness to participate in a pilot implementation of our solution.

We use a **Questionnaire**, You can view it form here: **view the questionnaire**

And here is it a sample of the answers we received:

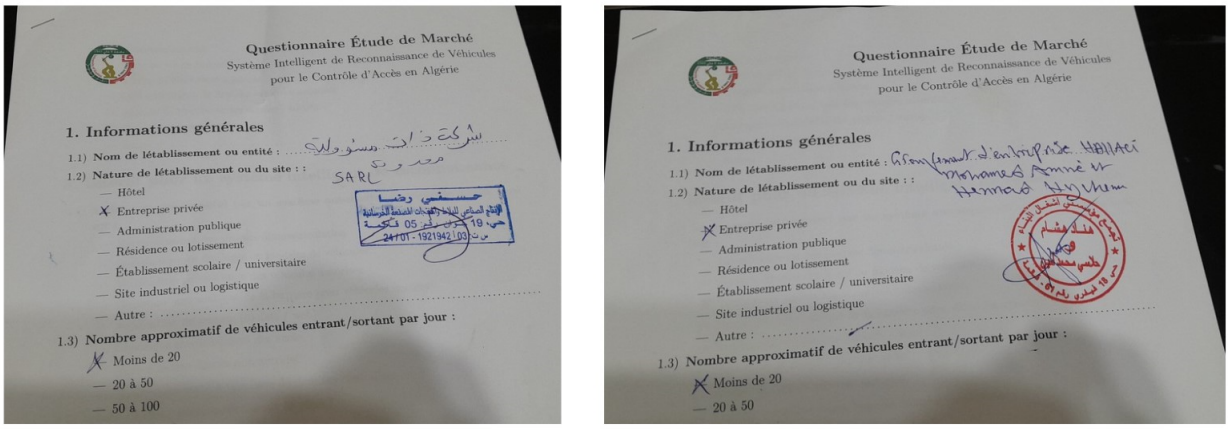


Figure 46: A sample of received answers.

H.1 Market Interest Statistics

- This figure shows the breakdown of participating organizations in the survey. All respondents were from private companies, representing different sectors.

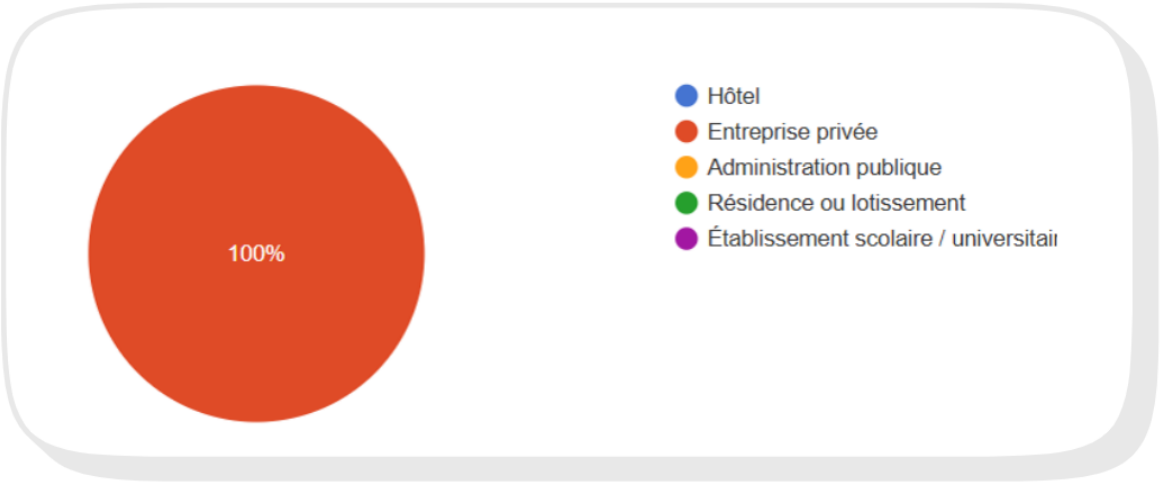


Figure 47: Distribution of Respondent Organization Types.

- This figure presents the frequency of vehicle access per day across the surveyed establishments.

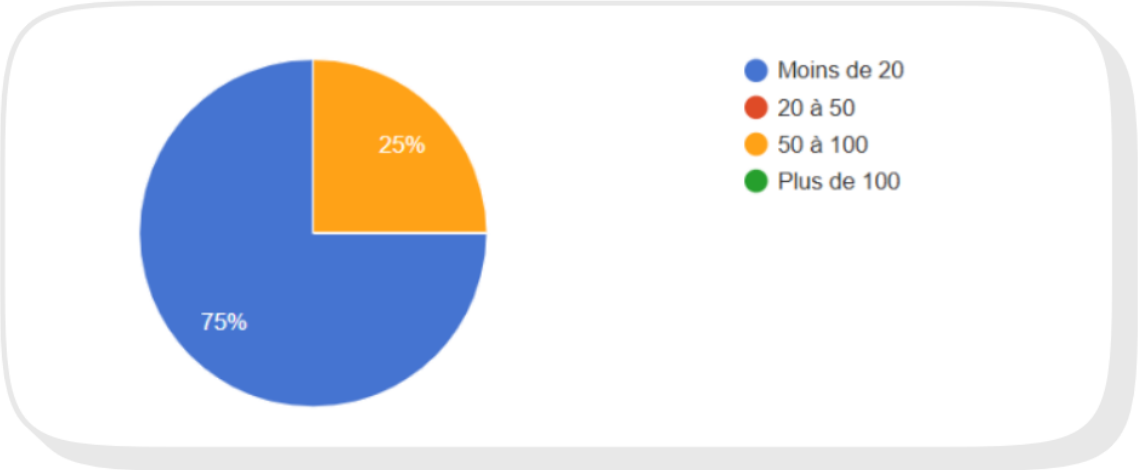


Figure 48: Daily Vehicle Access Volume Distribution.

- This figure presents the percentage of surveyed organizations that currently have an access control system in place versus those that do not

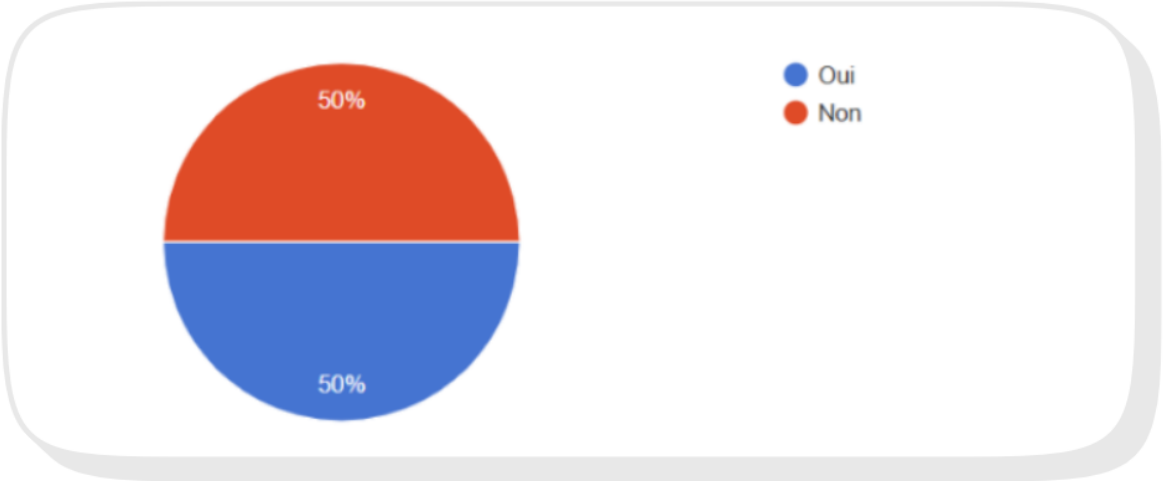


Figure 49: Prevalence of Existing Access Control Systems Among Respondents.

- This figure illustrates the different types of access control systems currently implemented among respondent organizations.

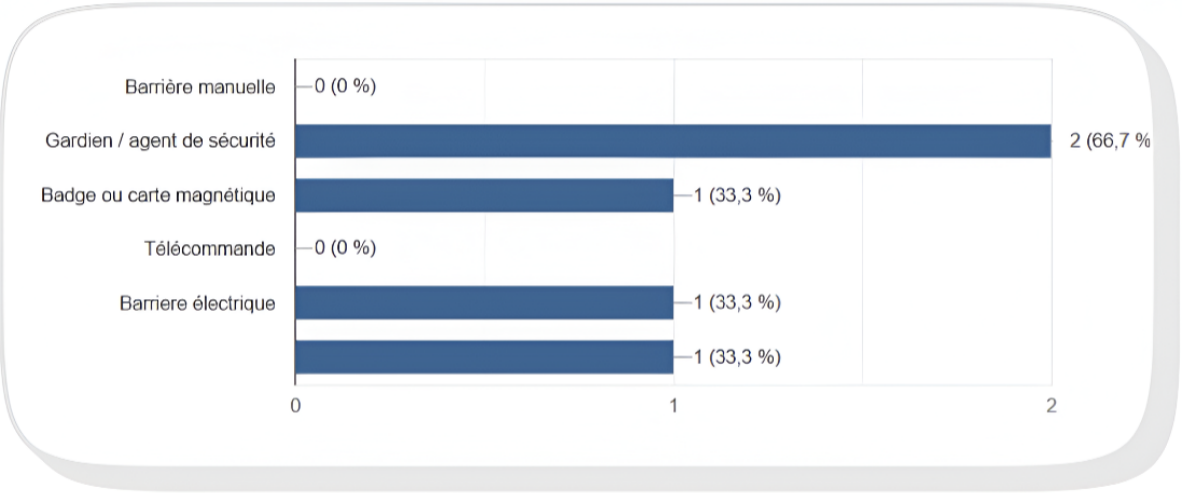


Figure 50: Current Access Control System Type Utilization.

- This figure identifies and ranks the most significant operational challenges faced by organizations regarding their access control systems.

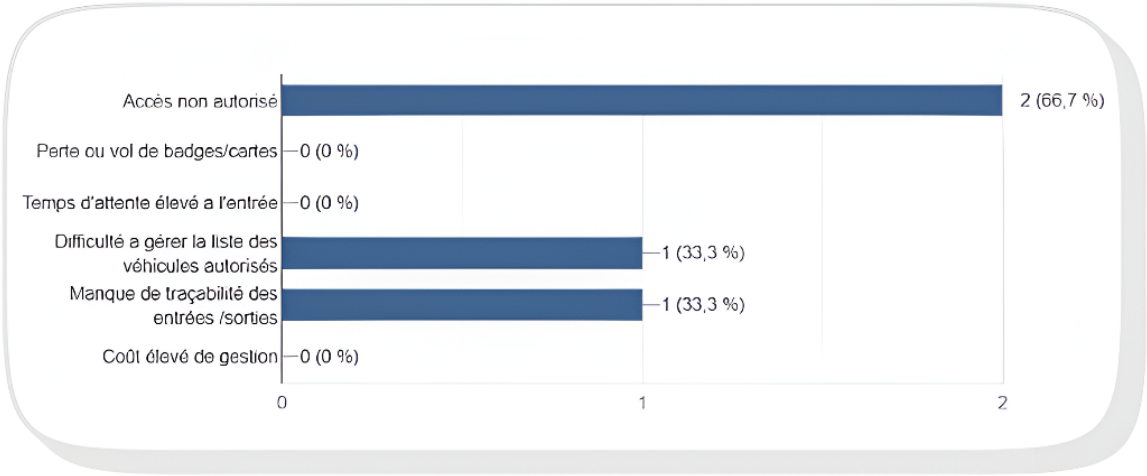


Figure 51: Operational Challenges in Current Access Management Systems.

- This figure presents respondent organizations' level of interest in adopting various automated access control technologies

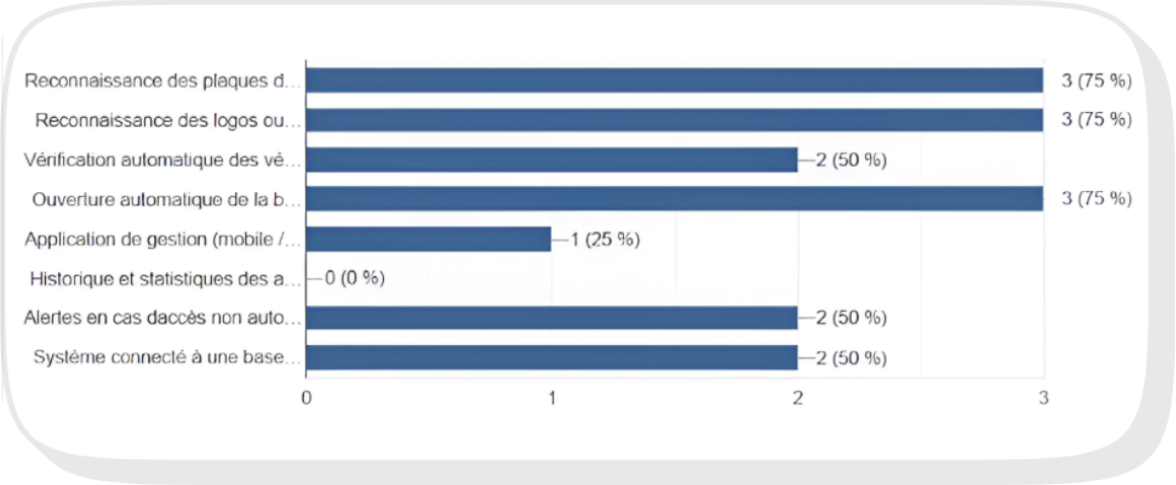


Figure 52: Statistical representation of automated access control system adoption interest.

- This figure presents survey data on organizations' openness to evaluating a prototype access control system.

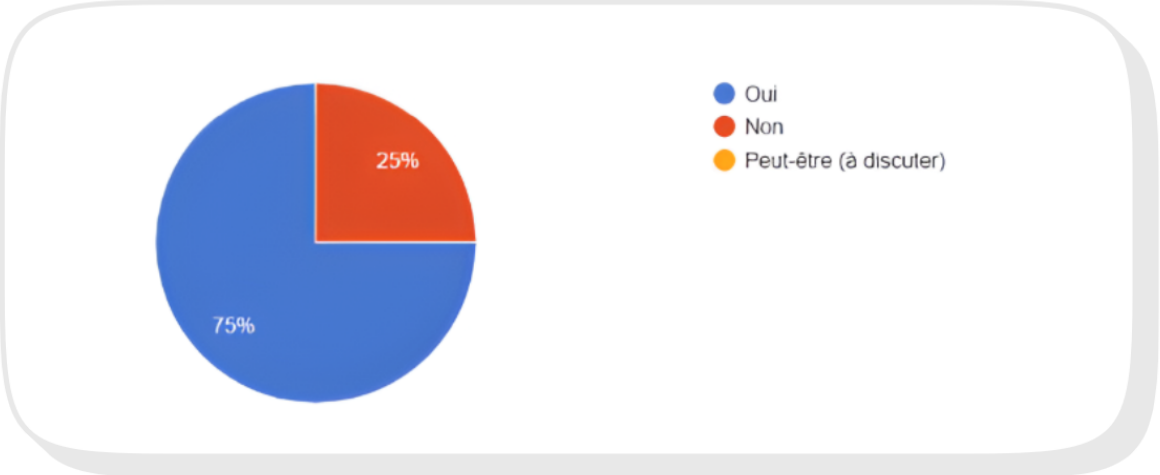


Figure 53: Respondents Interested in Prototype Evaluation Period.

THE BUSINESS MODEL CANVAS

KEY PARTNERS

-Government agencies (residence, univesities)
-Parking facility operators
-Hardware providers (cameras, sensors)
-AI/Computer vision technology providers
-Investors and public safety stakeholders

KEY ACTIVITIES

-Collected Algerian vehicle data
-Developed recognition algorithms
-Integrated core modules (plate, model, access)
-Designed and deployed system
-Piloted in key locations
-Built UI and dashboard
-Conducted outreach and marketing

KEY RESOURCES

AI system
Skilled tech team
Training datasets
Institutional partnerships

VALUE PROPOSITIONS

Automates manual surveillance tasks
Reduces human error
Adaptable to Algerian context
Scalable/customizable system
Real-time tracking & alerts with activity logs.

CUSTOMER RELATIONSHIPS

Real-time verification system
Ongoing technical support
Custom setup for client sites

CHANNELS

Direct sales to institutions
System integrators
Online platform/demos
Tech exhibitions

CUSTOMER SEGMENTS

Public institutions
Private businesses
Residential complexes and gated communities
Smart city planners, transport authorities
Security entities

COST STRUCTURE

-AI/software development
-Cloud hosting and storage
-Human resources
-Marketing/communication
-Continuous technical support

REVENUE STREAMS

-Software licenses
-annual subscriptions
-Premium alerts
-Technical support contracts
-Client-specific features