People's Democratic Republic of Algeria

Ministry of Higher Education for 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

Algerian license plate detection and recognition using deep learning

Presented by: Mihoub Imane

Jury Members

Chairman: Pr. Kouahla Mohamed Nadjib

Supervisor: Dr. Bouressace Hassina

Examiner: Ms. Madi Leyla

June 2023

Acknowledgments

Alhamdulillah who made it possible for me. Alhamdulillah, for granting me the opportunity, strength, and capability to undertake this. I extend my heartfelt thanks to my supervisor, Dr.Bouressace Hassina whose unwavering support, wisdom, and guidance have been instrumental in shaping this dissertation. Their expertise, constructive feedback, and dedication have been invaluable, and I am truly grateful for their mentorship.

I am truly blessed to have an exceptional family my little family who has been my rock and source of unwavering support. To my loving father, caring mother, dear sister Wejdane , and the best brother Abdou, I cannot find the words to express my deep gratitude for everything you have done for me. Your love, encouragement, and unwavering presence in my life have given me the strength and motivation to pursue my dreams. Thank you for always being by my side, believing in me, and providing the support and guidance I needed. I am forever grateful for the love and support you have given me.

I am incredibly grateful to Ghada for being a sister to me, not by blood but by the connection of our souls. I consider myself fortunate to have her in my life. Ghada, thank you for always being there for me, providing unwavering support, and giving me the strength to persevere. I am deeply appreciative of our friendship and the bond we share. I also want to express my thanks to our university for bringing us together and giving us the opportunity to meet.

I am sincerely grateful to my loving and supportive big family, as well as those who genuinely care for me an loves me. Your unwavering support and well wishes have been a constant source of strength and motivation for me. I am truly fortunate to have you all in my life, encouraging me to be the best version of myself. Additionally, I want to acknowledge and thank myself for the determination and resilience I have demonstrated throughout my journey. Despite the challenges, I have persevered and grown. I will continue to move forward, embracing new opportunities, and reminding myself to never give up.

Abstract

License plate detection and recognition systems play a crucial role in various applications such as traffic surveillance, parking management, and law enforcement. In this paper, we propose a deep learning-based license plate detection and recognition system. We leverage the power of the YOLOv5 model for license plate detection, which provides efficient and accurate object detection capabilities. For Algerian license plate recognition, we employ a CNN model trained on a large dataset of labeled license plate images.

Through extensive experiments and evaluations, we achieve outstanding results. Our system achieves an impressive precision of 87% in license plate detection, accurately identifying license plates in diverse environmental conditions. Moreover, in the license plate recognition phase, our CNN model achieves a remarkable accuracy of 93%, successfully recognizing and extracting characters from the detected license plates.

The combination of YOLOv5 for efficient license plate detection and the CNN model for accurate recognition results in a robust and effective license plate detection and recognition system. The system's high precision in detection and accuracy in recognition make it suitable for real-world applications requiring reliable license plate analysis.

Keywords: License plate detection,License plate Recognition, ALPR, Yolov5, (CNN),Deep learning.

Résumé

Les systèmes de détection et de reconnaissance de plaques d'immatriculation jouent un rôle crucial dans diverses applications telles que la surveillance du trafic, la gestion du stationnement et l'application de la loi. Dans cet article, nous proposons un système de détection et de reconnaissance de plaques d'immatriculation basé sur l'apprentissage profond. Nous exploitons la puissance du modèle YOLOv5 pour la détection des plaques d'immatriculation, qui offre des capacités de détection d'objets efficaces et précises. Pour la reconnaissance des plaques d'immatriculation algériennes, nous utilisons un modèle de réseau de neurones convolutif (CNN) entraîné sur un grand ensemble de données d'images de plaques d'immatriculation annotées.

Grâce à des expérimentations et évaluations approfondies, nous obtenons des résultats exceptionnels. Notre système atteint une précision impressionnante de 87% dans la détection des plaques d'immatriculation, identifiant avec précision les plaques d'immatriculation dans des conditions environnementales diverses. De plus, dans la phase de reconnaissance des plaques d'immatriculation, notre modèle CNN atteint une précision remarquable de 93%, reconnaissant et extrayant avec succès les caractères des plaques d'immatriculation détectées.

La combinaison de YOLOv5 pour une détection efficace des plaques d'immatriculation et du modèle CNN pour une reconnaissance précise permet d'obtenir un système de détection et de reconnaissance de plaques d'immatriculation robuste et efficace. La haute précision de détection et la précision de reconnaissance du système le rendent adapté à des applications réelles nécessitant une analyse fiable des plaques d'immatriculation.

Mots clés: Détection de plaques d'immatriculation, Reconnaissance de plaques d'immatriculation, ALPR, Yolov5, CNN, Apprentissage profond.

Contents

General Introduction 1

1 Object detection and deep learning **3**
1.1 Introduction 3
1.2 Object detection 3
1.2.1 Models of object detection 3
1.2.2 Object detection application 6
1.3 Deep Learning 9
1.3.1 Deep Learning Methods 9
1.4 Conclusion 13

2 State-of-the-Art **14**
2.1 Introduction 14
2.2 Automatic license plate detection and recognition 14
2.3 License plate detection 14
2.3.1 Edge detection 14
2.3.2 Texture-Based Methods 16
2.3.3 Convolutional neural network CNN 17
2.3.4 You-only-look-once (YOLO) 17
2.4 Character Segmentation 19
2.4.1 Character Segmentation Using Pixel Connectivity 19
2.4.2 Character segmentation using projection profiles 19
2.4.3 Character segmentation using deep learning 20
2.5 License plate recognition 20
2.5.1 Convolutional Neural Network 20
2.5.2 Recurrent Neural Network (RNN) 21
2.5.3 Optical Character Recognition (OCR) 22
2.5.4 K-means Clustering-based Approach 23
2.5.5 Template matching 23
2.6 Conclusion 23

3 Conception **24**
3.1 Introduction 24
3.2 System goals 24
3.3 Characteristics of Algerian plates 24
3.4 Architecture of the system 25
3.5 License plate detection 26
3.5.1 Yolov5 model learning 26
3.5.2 License plate detection 28

3.5.3 Extract the license plate . . . 28
3.6 License plate recognition . . . 28
3.6.1 Preprocessing of extracted image . . . 28
3.6.2 Character segmentation . . . 33
3.6.3 Convolutional Neural Network (CNN) learning model . . . 34
3.6.4 Character recognition . . . 35
3.6.5 Classification . . . 35
3.7 Conclusion . . . 36

4 Implementation **37**
4.1 Introduction . . . 37
4.2 Environment . . . 37
4.3 Programming language . . . 37
4.3.1 Python . . . 37
4.3.2 Jupyter notebook . . . 38
4.4 Libraries . . . 38
4.4.1 Tensorflow . . . 38
4.4.2 Keras . . . 38
4.4.3 PyTorch . . . 38
4.4.4 OpenCv . . . 38
4.4.5 Matplotlib . . . 39
4.4.6 NumPy . . . 39
4.4.7 scikit-image (skimage) . . . 39
4.4.8 Tkinter . . . 39
4.4.9 Customtkinter . . . 40
4.4.10 PIL (Python Imaging Library) . . . 40
4.5 System overview . . . 40
4.6 Usage scenario . . . 41
4.7 Results analysis . . . 45
4.7.1 Results of detection model . . . 45
4.7.2 Results of recognition model . . . 47
4.8 Discussion . . . 48
4.9 Conclusion . . . 49
General Conclusion . . . 50

List of Figures

1.1 Architecture of R-CNN 4
1.2 Architecture of SPP-net 4
1.3 Architecture of YOLO 5
1.4 Architecture of SSD 6
1.5 Example of medical imaging 6
1.6 Example of robotic 7
1.7 Example of facial expression detection 8
1.8 Pictures of real-life applications of ALPR system: ALPR system in (a) traffic law enforcement, (b) automatic toll collection, and (c) parking automation. . . 9
1.9 Basic convolutional neural network architecture 10
1.10 Example of max pooling and average pooling operations 11
1.11 Basic Recurrent neural network architecture. 12
1.12 A deep neural network architecture 13

2.1 Basic organization of an ALPR system. 15
2.2 CNN architecture for the MD-YOLO model 19

3.1 Algerian license plate. 24
3.2 The architecture of the proposed license plate recognition system. 25
3.3 The Basic Phases of YOLOv5 Model. 26
3.4 Example Of Gray-Scale License Plate. 29
3.5 Example Enhanced License Plate. 30
3.6 Example of blurred image. 30
3.7 Example of Otsu's thresholding license plate 31
3.8 Example 1 of morpholoical operation. 32
3.9 Example 2 of morpholoical operation. 32
3.10 Example of skew correction. 33
3.11 Example of classification. 36

4.1 The home interface of our system. 40
4.2 Open an image. 41
4.3 License plate detetion. 42
4.4 License plate extration. 42
4.5 License plate pre_processed. 43
4.6 Character segmentation. 43
4.7 Character Recognition. 44
4.8 Character Classification. 44
4.9 Example of non-Algerian plate. 45
4.10 The training results of YOLOv5. 46
4.11 The output of running model on a test image. 46

4.12 Accuracy Curve. 47
4.13 Training Loss Curve. 47
4.14 Correct vs incorrect characters recognized 48

List of Tables

4.1 Characteristics of the material used. 37
4.2 The training result statistics. 45

General Introduction

In today's world, there is a growing demand for automatic license plate detection and recognition systems. These systems offer numerous benefits, including enhanced security and law enforcement, improved traffic management and surveillance, streamlined parking systems and access control, efficient toll collection and electronic payment processes, and support for smart city initiatives. With advancements in deep learning techniques, these systems have become more accurate and efficient, making them indispensable for addressing contemporary challenges and ensuring public safety and convenience.

Taking into account that each country has its specific license plate, these differences can manifest in various forms, including plate types and plate structures. In the case of Algerian license plates, they have their unique design and formatting rules that must be considered. The Algerian license plate system consists of a combination of numbers arranged in a particular sequence. Moreover, the color scheme and overall layout of the plate also contribute to its distinctiveness. Creating a powerful system that can accurately recognize and process the diverse range of Algerian license plate variations poses a significant challenge. The system needs to be adaptable enough to handle changes in font styles, plate sizes, and any future modifications to the Algerian license plate format. To achieve this, extensive research and data collection on Algerian license plates are necessary. The system should be trained using a large dataset of genuine Algerian license plates, encompassing various styles and variations. Additionally, it should be equipped with advanced image processing techniques and algorithms to accurately recognize and extract information from the license plate images. By developing a comprehensive and adaptable system that can handle the diversity of Algerian license plates, it would contribute to improved efficiency in traffic enforcement, vehicle identification, and overall road safety in Algeria.

In this thesis, we propose a comprehensive system for Algerian license plate detection and recognition that can handle the diverse range of plate variations with utmost efficiency. Our work is organized into four chapters, each focusing on key aspects of the system development and evaluation.

First Chapter: Dedicated to the exploration of object detection and deep learning techniques. We provide an in-depth analysis of various object detection algorithms, such as Faster R-CNN, YOLO, and SSD, emphasizing their strengths and limitations in the context of license plate detection.

Second Chapter: Dedicated to exploring the state-of-the-art techniques in license plate detection and recognition. We conduct an extensive review of existing methods, including deep learning-based approaches, traditional computer vision techniques, and hybrid models.

Third Chapter The third chapter focuses on the conception of our system. In this chapter, we delve into the initial stages of the system development process.

Fourth Chapter The fourth chapter typically delves into the implementation phase of the system development project. Introduces working environment and the results obtained from evaluating the performance of the algorithms.

Chapter 1

Object detection and deep learning

1.1 Introduction

Deep learning has revolutionized the field of computer vision by providing the ability to learn complex features directly from raw image data. CNNs, a specific type of deep learning model, have shown remarkable success in a wide range of computer vision tasks, including object detection. Recently, there has been an explosion of research in object detection using deep learning, such as region-based convolutional neural networks (R-CNNs), single shot multibox detectors (SSDs), and you only look once (YOLO) models.
This chapter provides an overview of object detection techniques, with a focus on deep learning-based approaches, and discusses the different types of deep learning models used for object detection.

1.2 Object detection

Object detection is an important area in image processing and computer vision. It involves using deep learning and computer vision techniques to identify various types of objects in video streams and files. While humans can effortlessly recognize objects in the real world, machines require advanced algorithms to perform this task. Object detection is a critical aspect of computer vision, and it involves locating and identifying specific objects in an image. Object recognition is a fundamental task in computer vision that involves finding and identifying objects in digital images, as well as in stored and real-time videos. In other words, object recognition is the process of determining the identity of an object being observed in an image or video .[51]

1.2.1 Models of object detection

Region-based Convolutional Neural Networks RCNN

The RCNN (Region-based Convolutional Neural Network) works by first extracting a set of object proposals using selective search. These object candidate boxes are then rescaled to a fixed size image and passed through a CNN model, such as AlexNet that has been pre-trained on ImageNet to extract features. The features are then fed to linear SVM classifiers that predict the presence of an object within each region and recognize object categories. (see Figure 1.1)

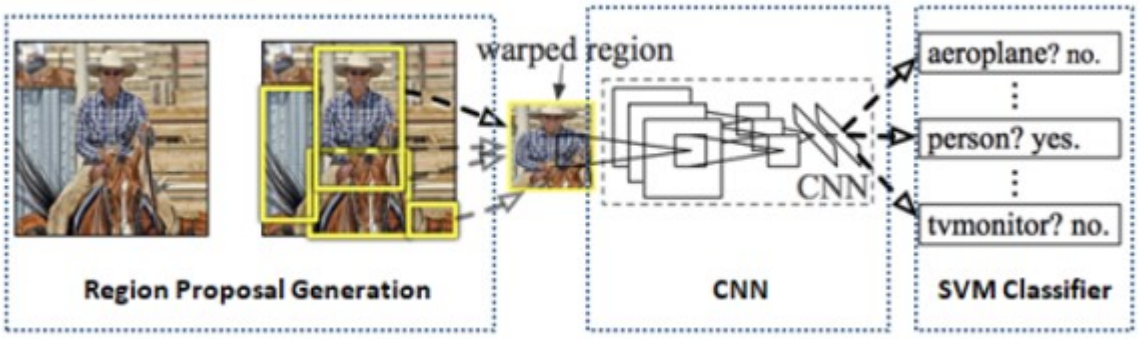

Figure 1.1: Architecture of R-CNN
[47]

RCNN has shown significant improvement in mean Average Precision (mAP) on VOC07 compared to DPM-v5 . However, it has a major drawback of redundant feature computations on a large number of overlapped proposals, leading to a slow detection speed of around 14 seconds per image with GPU.
To solve this problem, SPPNet (Spatial Pyramid Pooling Network) was proposed later in the same year. [63]

Spatial Pyramid Pooling (SPP-net)

SPP-Net is an improved version of the RCNN that offers faster speed. It introduces a spatial pyramid pooling (SPP) layer that eliminates the restriction on network fixed size. Unlike RCNN, SPP-Net only runs the convolution layer once for the whole image, regardless of size, and then uses the SPP layer to extract features. This approach helps avoid repeat convolution operations on the candidate area, reducing the number of convolution times required.(see figure 1.2) [61]

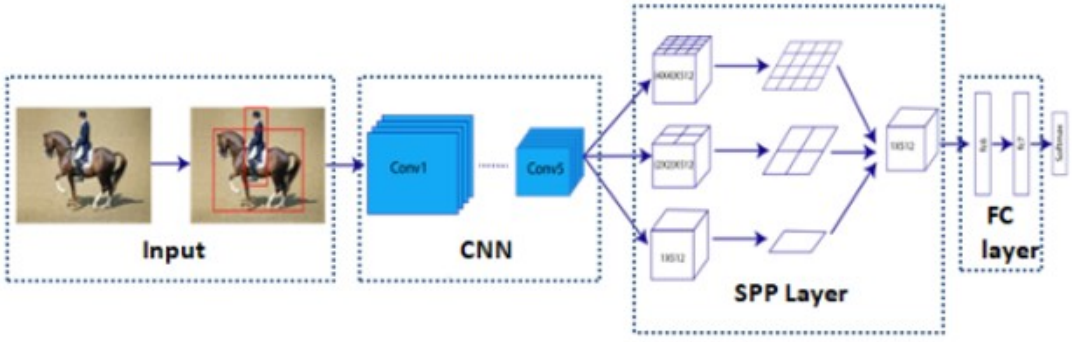

Figure 1.2: Architecture of SPP-net
[47]

Compared to RCNN, SPP-Net calculates the convolution on the Pascal VOC 2007 dataset by 30-170 times faster, and its overall speed is 24-64 times faster than the RCNN. This improved speed is achieved because SPP-Net eliminates the need for redundant feature computations on a

large number of overlapped proposals, making it more efficient and faster for object detection. [61]

You Only Look Once (Yolo)

YOLO was introduced in 2015 by R. Joseph et al. and was the first one-stage detector in the deep learning era. [8] YOLO is known for its speed and can run at 155fps with a VOC07 mAP of 52.7% . An enhanced version of YOLO can run at 45fps with a VOC07 mAP of 63.4%. YOLO uses a different approach from two-stage detectors as it applies a single neural network to the full image. This network divides the image into regions and simultaneously predicts bounding boxes and probabilities for each region. (see figure 1.3) [63]

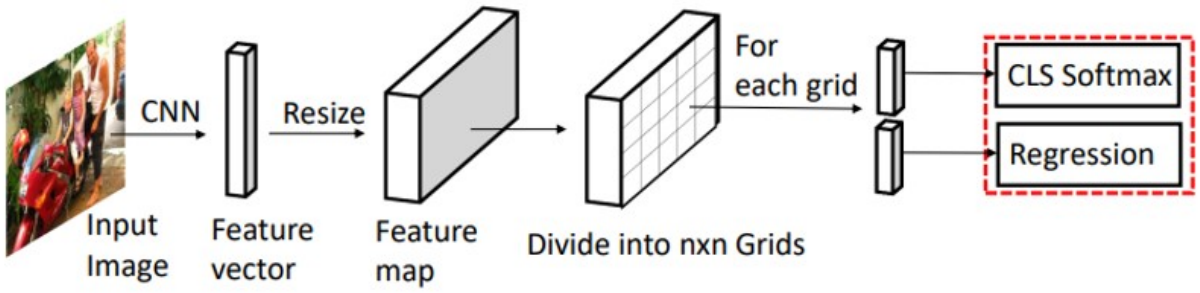

Figure 1.3: Architecture of YOLO
[57]

Although YOLO has made significant improvements in detection speed, it suffers from a drop in localization accuracy compared to two-stage detectors, especially for small objects. Subsequent versions of YOLO and the proposed SSD have focused on this problem. Recently, YOLOv7, a follow-up work from the YOLOv4 team, was proposed. It outperforms most existing object detectors in terms of speed and accuracy (ranging from 5 FPS to 160 FPS) by introducing optimized structures like dynamic convolution and cross-stage partial connections.[63]

Single Shot MultiBox Detector (SSD)

The single shot multibox detection (SSD) model is an object detection algorithm that takes an entire image as input and uses multiple convolutional layers with varying filter sizes (10×10, 5×5, and 3×3) to extract feature maps at different scales as showing in figure 1.4 . These feature maps are then used to predict bounding boxes for objects in the image.[6] To refine the bounding box predictions, SSD uses extra feature layers with 3×3 filters. The anchor box of SSD is similar to the default box of Fast R-CNN and has parameters that include the center coordinates, width, and height.
To handle objects of different scales, SSD predicts bounding boxes after multiple convolutional layers. Since each convolutional layer operates at a different scale, the model can detect objects of various sizes. (see figure 1.4)
[47]

Finally, SSD produces a vector of probabilities for each class of object, indicating the confidence level of the predicted bounding box. This approach achieves a good balance between accuracy and speed, making SSD a popular choice for object detection in various real-world applications. [47] .

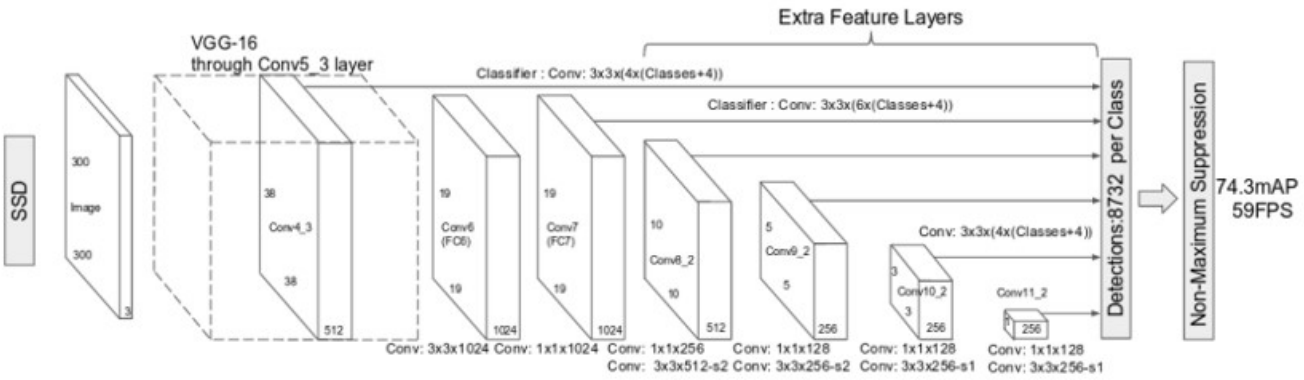

Figure 1.4: Architecture of SSD
[47]

1.2.2 Object detection application

Object detection has a wide range of applications in computer vision and artificial intelligence, including:

Medical imaging

In medical imaging, object detection is a crucial tool in assisting healthcare providers in identifying and diagnosing various conditions. The technique is used to detect and segment various structures within medical images such as tumors, organs, or blood vessels. The output from the object detection algorithms provides valuable information that helps healthcare providers make informed decisions about patient treatment. By detecting these objects in medical images, object detection in medical imaging saves time, reduces human error, and helps improve the accuracy of diagnoses.[51]

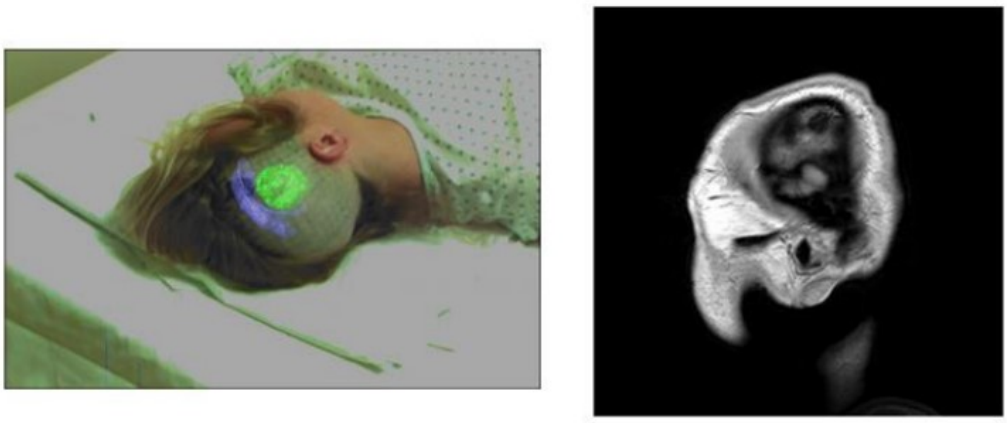

Figure 1.5: Example of medical imaging
[51]

Robotic

This requires autonomous assistive robots to have the capability to efficiently process visual information in real-time, enabling them to quickly adapt and respond to changes in their environment. Object detection and recognition is a crucial first step in achieving this goal and is necessary for the robots to function effectively .[51]

Figure 1.6: Example of robotic
[51]

Facial expression detection

Human faces provide a wealth of information that can help in understanding the user's current state of mind, making it a valuable tool for constructing effective human-computer interactions. Over the past decade, facial expression classification and recognition have been extensively studied, resulting in a wide range of applications. These include socially intelligent robots with emotional intelligence, assistance systems for autistic individuals, emotion detection systems for the disabled, pain or stress detection systems for psychological studies, intelligent tutoring systems, and portable mobile applications that can automatically insert emotions in chat applications.[12]

To be effective, facial expression detection systems must be capable of real-time operation. As these expression recognition systems become more robust and able to work in real-time, they will open up new avenues for innovation and lead to many other applications. [12]
Overall, the research and development of facial expression detection systems have significant potential to improve human-computer interactions and have a positive impact on various areas, including healthcare, education, and social interactions. (see figure 1.7)[12]

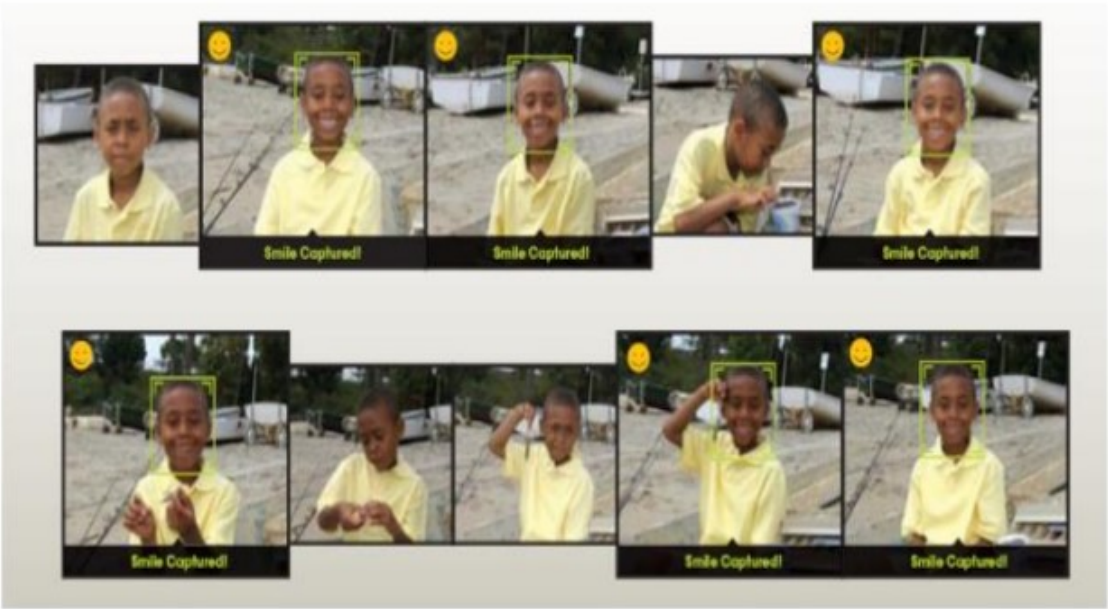

Figure 1.7: Example of facial expression detection
[51]

License plate detection

Automatic License Plate Recognition (ALPR) systems are becoming increasingly popular because of their wide range of applications in intelligent transportation systems, such as traffic law enforcement, traffic monitoring, vehicle park management, toll collection, and security control in restricted areas like military campsites and protected sanctuaries. These systems are designed to prevent fraud and intensify security in specific areas, such as searching for missing vehicles or vehicles related to crimes. Without ALPR systems, such tasks would require a significant amount of labor, time, and resources. Additionally, manual intervention in such tasks may lead to erroneous interpretations, and it is practically challenging for a human to remember or read a license plate of a moving vehicle efficiently.(see figure 1.8)[43]

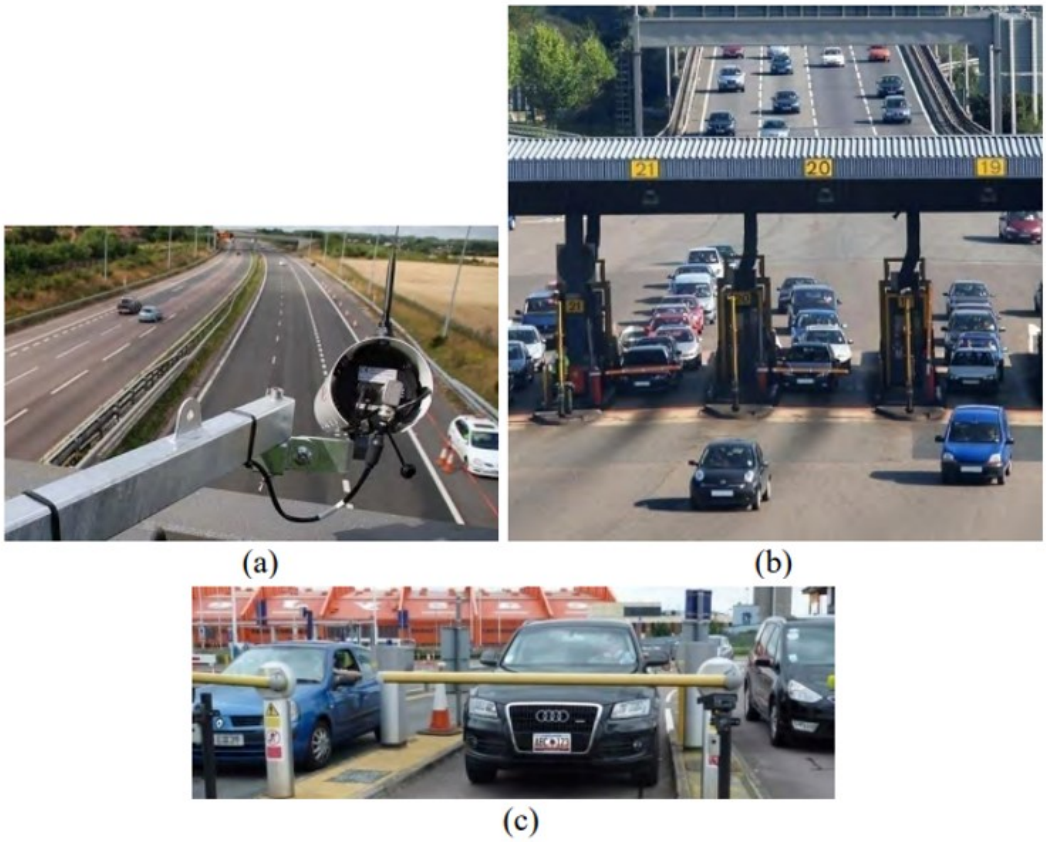

Figure 1.8: Pictures of real-life applications of ALPR system: ALPR system in (a) traffic law enforcement, (b) automatic toll collection, and (c) parking automation.
[6]

1.3 Deep Learning

Deep learning is a subset of machine learning that involves building and training neural networks with multiple layers to learn from large amounts of data, in order to make predictions or decisions on new data. It has been successfully applied to a wide range of tasks, including image and speech recognition, natural language processing, and recommendation systems.

1.3.1 Deep Learning Methods

Convolutional Neural Network (CNN)

The Convolutional Neural Network (CNN) was originally introduced by LeCun [16] in 1998 and is a type of neural network that incorporates the concepts of receptive fields, weight sharing, and subsampling. It is a special kind of multilayer perceptron that is trained using supervised learning and backpropagation. CNN has proven to be highly successful in computer vision and has achieved state-of-the-art results in tasks such as character recognition, object recognition, face detection, pose estimation, speech recognition, license plate recognition, as well as image preprocessing and segmentation tasks.[5]

A convolutional neural network (CNN) typically consists of three main types of layers: convolutional layers, pooling layers (also called subsampling layers), and fully-connected layers as showing in image below, these layers work together to learn features and make predictions on input data, such as images. (see figure 1.9)[60]

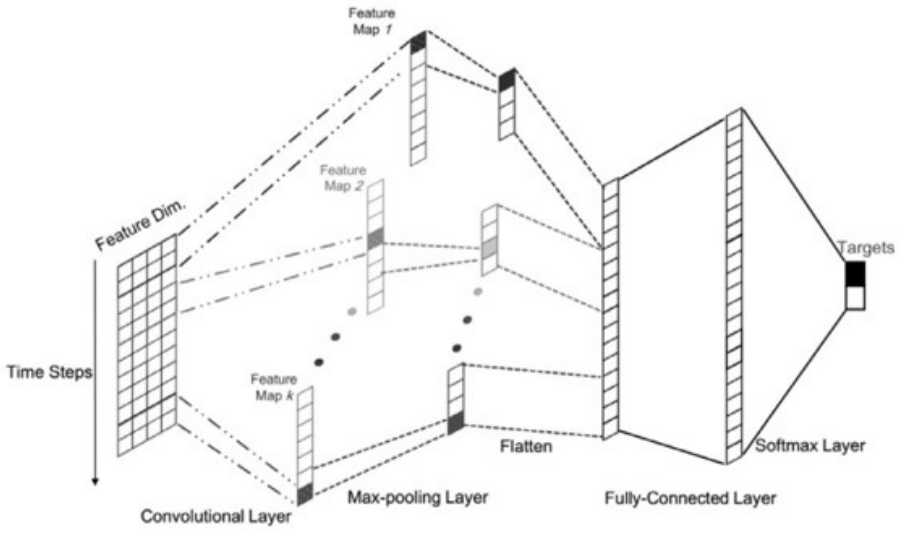

Figure 1.9: Basic convolutional neural network architecture
[60]

Convolutional layer The Convolutional Neural Network (CNN) utilizes convolutional layers to extract features from input images. Neurons in each convolutional layer are arranged into feature maps, where each neuron has a receptive field connected to a neighborhood of neurons in the previous layer through trainable weights, also known as a filter bank. The input image is convolved with the learned weights to generate a new feature map, which is then passed through a nonlinear activation function denoted by f(·). The k-th output feature map Y_k is computed using the formula:

$$Y_k = f\left(\sum_{i=1}^{n} W_{ki} \cdot X_i + b_k\right)$$

where x represents the input image, Wk is the convolutional filter related to the kth feature map, and represents the 2D convolutional operator that calculates the inner product of the filter model at each location of the input image. All neurons within a feature map have equal weights, while different feature maps within the same convolutional layer have different weights to extract multiple features at each location. The use of nonlinear activation functions such as the rectified linear unit (ReLU) enables the extraction of nonlinear features from the feature maps. The ReLU function has gained popularity due to its success in improving the performance of CNNs. Further research focuses on the development and application of novel activation functions to enhance the performance of CNNs. [38]

pooling layers After the convolutional layer detects features in an image, their exact location becomes less important, as what matters more is their relation to the neighboring features. Subsampling, achieved through the subsampling layer, enhances translational invariance by reducing spatial resolution and sensitivity to shifts and distortions.

Pooling is used to further subsample the feature maps by sliding a window across them and computing a single output value per window, thus reducing the size of the feature maps. The size of the window used for pooling can be adjusted based on the requirements of the model.[5] Two pooling operations are defined below:

A verage pooling The average pooling operation computes the average value of the elements within a receptive field, placed at each position in the input feature map. The resulting value is then propagated to the corresponding location in the output feature map.[5]

Max pooling Max pooling involves taking the maximum value from each sub-region of the activation map and using it to create a new matrix. This limits the number of learnable features while preserving important features of the image. Typically, a 2x2 filter is used for max pooling. (see figure 1.10)[2]

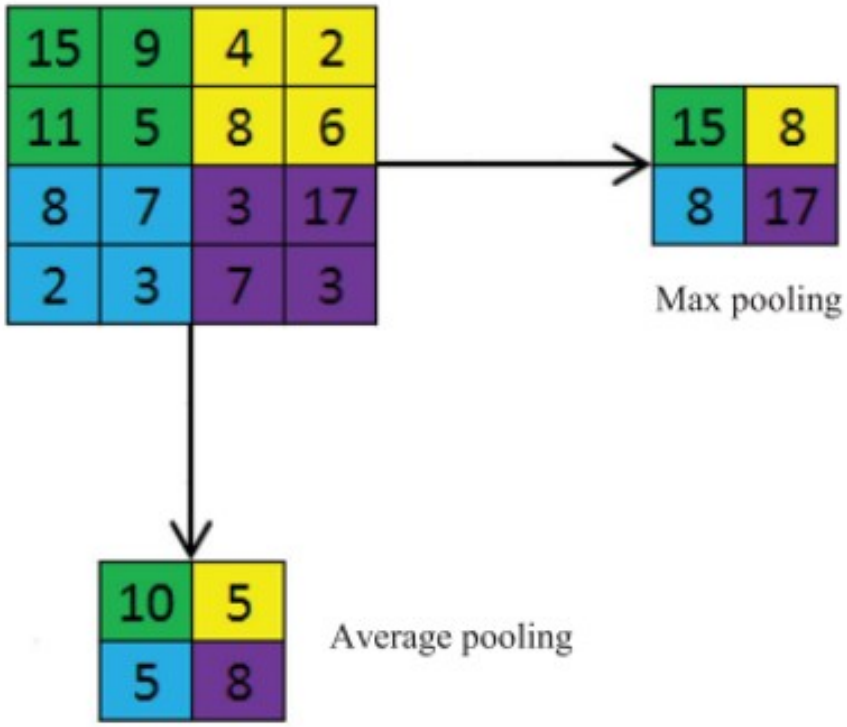

Figure 1.10: Example of max pooling and average pooling operations
[38]

Fully-connected layers At the end of a CNN there are one or more fully connected layers (every node in the first layer is connected to every node in the next layer).They consist in performing a classification based on the features extracted from the convolutions. The final layer contains a Softmax activation function, which generates a probability value from 0 to 1 for each of the class labels that the model attempts to predict. In some recent CNNs network architectures, the fully connected layers can be replaced by several average pooling layers. This allows these networks to significantly reduce the total number of parameters and which allows better

prevention of overfitting. [30]

Recurrent Neural Network (RNN)

The RNN is a popular algorithm in deep learning, particularly for NLP and speech processing, that utilizes sequential information in the network. This allows for the use of embedded structures in data sequences to convey useful knowledge, such as understanding the context of a word in a sentence. An RNN includes an input layer, hidden (state) layer, and output layer and can be seen as short-term memory units. A deep RNN, which takes advantage of a deeper network and reduces difficult learning in deep networks, can include three approaches: deep "Input-to-Hidden," "Hidden-to-Output," and "Hidden-to-Hidden". One issue with RNNs is their sensitivity to vanishing and exploding gradients, which can cause the network to forget initial inputs with the entrance of new ones. To address this issue, Long Short-Term Memory (LSTM) is used to provide memory blocks in its recurrent connections with gated units to control information flow. Residual connections in very deep networks can also alleviate the vanishing gradient issue. (see figure 1.11)[34]

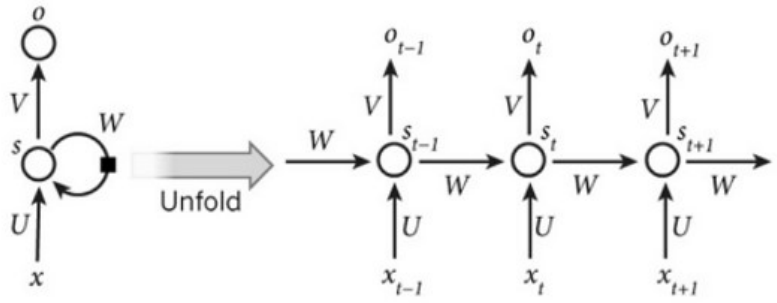

Figure 1.11: Basic Recurrent neural network architecture.
[60]

Deep Neural Network (DNN)

Deep Neural Networks (DNNs) are a type of machine learning model that are inspired by the structure and function of the human brain. They consist of multiple layers of interconnected nodes, also known as neurons, which process and transform input data to produce an output. Each layer in a DNN is responsible for learning and extracting different features from the input data, with the final layer producing the output.
DNNs are called "deep" because they typically have many layers, which allows them to learn complex representations of data. The number of layers in a DNN can range from a few to hundreds or even thousands, depending on the complexity of the task at hand.
One of the key advantages of DNNs is their ability to automatically learn features from raw input data without requiring manual feature engineering. This makes them particularly useful for tasks such as image recognition, where traditional machine learning models would require hand-crafted features to achieve good performance.
Overall, DNNs have become an important tool in modern machine learning and have been used to achieve state-of-the-art performance in a wide range of applications.[26]

The general design of a DNN is shown in figure 1.12 below.

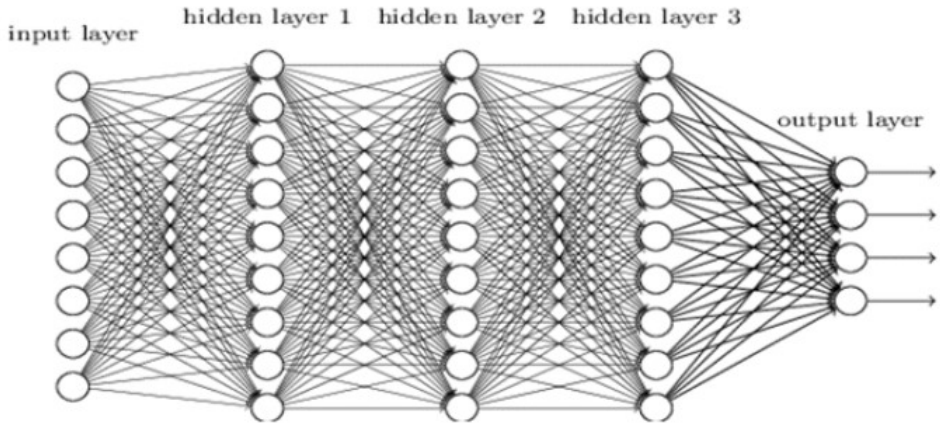

Figure 1.12: A deep neural network architecture
[29]

1.4 Conclusion

In conclusion, deep learning methods have revolutionized the field of object detection by providing high accuracy and efficiency. These methods include popular algorithms such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), as well as newer architectures such as YOLO and SSD. Object detection has a wide range of applications, including automatic license plate detection, and continues to be an active area of research with new advancements and techniques being developed.

Chapter 2

State-of-the-Art

2.1 Introduction

License plate detection and recognition is a critical task in many transportation-related applications, such as traffic monitoring, toll collection, and parking management. Traditional methods rely on image preprocessing, feature extraction, and classification, while deep learning-based approaches have shown great potential. CNNs, a specific type of deep learning model, have been widely used to learn complex features directly from raw image data. This chapter provides an overview of the techniques used for license plate detection and recognition, and reviews the traditional image processing techniques and recent advances in deep learning-based approaches for license plate detection and recognition.

2.2 Automatic license plate detection and recognition

ALPR stands for Automatic License Plate Detection and it refers to the process of identifying and extracting the license plate area of a vehicle from an image without any human intervention. This is a crucial step in Automatic License Plate Recognition (ALPR) System which has three major stages, including image acquisition, LP area detection, and character feature extraction and recognition. The detection stage is particularly important as it needs to handle various challenges such as different LP locations, multiple LPs in the image, various LP sizes and colors, camera tilting and poor lighting, among others. Due to these challenges, ALPR is considered a difficult research topic in the field of image processing.The general processes involved in ANPR systems is shown in Figure 2.1. [5]

2.3 License plate detection

2.3.1 Edge detection

The algorithm proposed In [55]for vehicle number plate detection is based on the Euler number of a binary image and the Mexican hat operator for edge detection. While they claim a high success rate of 94-99% and an average accuracy of 96.17%, there is a situation where their algorithm can fail. If the license plate is black, the edge detection system may fail to properly recognize the edges of the plate. This is because the Mexican hat operator used for edge detection is a high-pass filter that amplifies the high-frequency components of the image. In the case of a black license plate, the edges may not have sufficient high-frequency components to

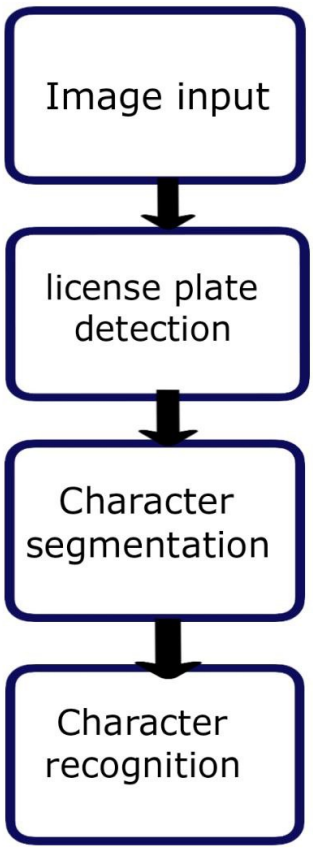

Figure 2.1: Basic organization of an ALPR system.
[43]

be detected by the operator, leading to a failure of the edge detection system.
To overcome this limitation, one approach could be to use a different edge detection algorithm that is more suitable for detecting edges in low-contrast images. For example, the Canny edge detection algorithm is a popular choice for edge detection in computer vision applications and is known to perform well in low-contrast images.
Alternatively, the algorithm could be modified to incorporate additional features or information to help detect the edges of black license plates. For example, the algorithm could use color information to distinguish between the black background and the white characters on the license plate, and then use this information to locate the edges of the plate. Additionally, the algorithm could use prior knowledge about the size and shape of license plates to help identify the region of interest and improve the accuracy of the edge detection process.[55]
The license plate detection system proposed in [10] is based on an enhanced Prewitt arithmetic operator and uses a projection method to locate the position of the license plate. The system is designed to work under various backgrounds and lighting conditions and is capable of detecting license plates in real-time.
The enhanced Prewitt arithmetic operator is a type of edge detection algorithm that is designed to improve the accuracy of edge detection in low-contrast images. The algorithm uses a combi-

nation of horizontal and vertical edge detection filters to identify the edges of the license plate. Once the edges of the license plate are detected, the projection method is used to determine the position of the license plate. The projection method works by projecting the edges of the license plate horizontally and vertically to identify the top and bottom edge areas along the edge. By analyzing the location of these edge areas, the system can accurately determine the position of the license plate and extract the license plate number.
According to Chen, their proposed system achieved a precision of 96.75%, indicating a high level of accuracy in license plate detection. Furthermore, they claim that their system is efficient enough to work in real-time, meaning that it can process video streams and detect license plates as the vehicle moves.
Overall, the license plate detection system proposed by Chen offers a promising solution for license plate detection under various backgrounds and lighting conditions. By using an enhanced Prewitt arithmetic operator and a projection method, their system achieves high accuracy and real-time efficiency, making it suitable for practical applications such as traffic monitoring and law enforcement.[10]

2.3.2 Texture-Based Methods

In [54], a Wavelet transform-based method for license plate detection is described. The method involves using the HL sub-band of the Wavelet transform for feature extraction, and then using the LH sub-band to verify the presence of a horizontal line around the feature. The authors report an accuracy of 97.33% for the localization process using this method.
One advantage of texture-based methods, such as the one described in [54], is that they are robust against license plate deformation. This means that the method can still accurately detect and recognize license plates even if they are bent or distorted. This is an important advantage since license plates can become deformed due to various reasons such as damage or natural wear and tear.
However, a drawback of texture-based methods is that they can involve complex computations and may not work well with complex backgrounds or different illumination conditions. For example, if the license plate is located in a scene with a cluttered background or poor lighting conditions, the texture features may be difficult to extract or may not be robust enough to accurately detect the license plate. Therefore, these methods may not be suitable for all scenarios and may require further optimization and customization to perform well in different conditions.
In [59], scan-line techniques were used for license plate detection. This method is based on the observation that the complexity of the plate region in a grayscale image is not seen anywhere else in the image, meaning that it is a unique feature that can be used for detection. This method does not depend on the boundary details of the license plate, making it a robust approach to detect license plates in various scenarios.
Zunino et al. [64] presented a novel method for localization using Vector Quantization (VQ). This method considers the actual content of the license plate rather than just its features like edges and contrast, which have been considered in other studies. The authors reported a detection accuracy of 98% and tested the method in a real-time industrial application.
License plates can create inconsistencies in the texture of the input image, which can disclose their presence in the image. This is because the letters and numbers on the license plate have a unique texture and appearance that can be used to distinguish them from the surrounding background. By using texture-based approaches like VQ, it is possible to leverage these unique features to accurately detect and recognize license plates in various scenarios. However, as mentioned earlier, these methods may not be suitable for all scenarios and may require further

optimization and customization to perform well in different conditions.

2.3.3 Convolutional neural network CNN

The authors of [14] proposed a convolutional neural network (CNN) based framework for detecting vehicle number plates. Their approach improves upon existing methods for detecting blurred and obscure images and is designed to work effectively under various lighting conditions.
CNNs are a type of neural network that are commonly used in computer vision applications, including object detection and image classification. They are well-suited for tasks that involve processing large amounts of image data, making them an ideal choice for license plate detection. The authors of [14] trained their CNN on a large dataset of license plate images to learn the features that are most relevant for detecting license plates. By training the network on a diverse range of images, the CNN is able to learn to detect license plates under various lighting conditions and in different environments.
The accuracy obtained by their proposed method is reported to be around 100%. This indicates that their system is able to correctly identify license plates in all cases, achieving a perfect score on their test dataset. However, it is important to note that the accuracy reported on a test dataset may not necessarily translate to real-world performance. In practice, there may be additional challenges such as variations in lighting and environmental conditions that could affect the accuracy of the system.
Overall, the CNN based framework proposed by the authors of [14] offers a promising solution for license plate detection. By using a neural network approach, their system is able to learn to detect license plates under a wide range of conditions and achieves high accuracy on their test dataset.
In the study by Selmi et al. [41], a CNN-based localization method was proposed. The method involved two major steps - preprocessing and classification. In the preprocessing stage, the input image was processed to remove noise and extract finer details. In the classification stage, a CNN classifier was used to distinguish possible bounding boxes as either license plate or non-plate regions. The method was evaluated on the Caltech data set and achieved a recall accuracy of 93.8% and f-score accuracy of 91.3%.
In another study by Zou et al. [62], two CNNs - shallow CNN and deep CNN were trained end-to-end for license plate detection. The shallow CNN was used to reduce the computational cost by removing most of the background regions from the image, while the deep CNN was used to detect the license plate from the remaining regions. Finally, non-maximum suppression (NMS) was applied to locate the exact plate region. The experimental results showed above-average accuracy with less computational cost. Overall, these studies demonstrate the effectiveness of deep learning neural networks, particularly CNNs, for license plate detection. These methods have shown promising results and can be further improved for real-world applications.

2.3.4 You-only-look-once (YOLO)

The success of real-time object detectors such as You-only-look-once (YOLO) has inspired many recent studies on automatic license plate recognition (ALPR). YOLO is a state-of-the-art object detection algorithm that uses a single neural network to predict bounding boxes and class probabilities for detected objects in real-time.
Silva et al. in [45] used YOLOv2, a popular object detection network, to detect vehicles without any modifications or fine-tuning. They treated the network as a black box and merged the

two classes of cars and buses on the PASCAL-VOC dataset, ignoring other classes. In contrast, WPOD-NET was specifically designed to detect license plates in a variety of different distortions. It was inspired by three other deep learning techniques: YOLO, SSD (Single Shot Detector), and STN (Spatial Transformer Network). WPOD-NET regresses coefficients of an affine transformation that unwarps the distorted license plate into a rectangular shape resembling a frontal view. This allows for more accurate detection and recognition of the license plate.
Overall, both approaches highlight the power and flexibility of deep neural networks in object detection tasks, and demonstrate how they can be adapted and refined for specific use cases.

In [25], Min proposed a vehicle license plate location system using the latest YOLO-L model and pre-identification plate to accurately detect the location of the license plate. The proposed method comprises two parts. First, the k-means++ clustering algorithm was used to select the appropriate size and number of candidate boxes for the license plate. Next, the YOLOv2 network model and depth were modified to detect the license plate accurately. In addition, a pre-identification algorithm was employed to separate license plates from other objects.
The proposed system was tested on a dataset containing different types of license plates and achieved a precision and recall of 98.86%. The high accuracy achieved by the proposed method demonstrates the effectiveness of using the YOLO-L model and pre-identification algorithm for license plate detection. This system can be useful for various applications such as traffic management, law enforcement, and toll collection system Parvin.

In [58], Xie et al. presented an approach that improved upon the original YOLO framework for license plate detection. Their proposed framework, called MDYOLO, is able to handle multi-directional license plates by providing information about the angle of rotation in addition to the object's center coordinates, height, and width.
To achieve this, Xie et al. redesigned the CNN architecture of YOLO by adding an extra branch to the output layer for predicting the angle of rotation. The model was trained on a large dataset of labeled license plate images captured under various lighting conditions and with different degrees of occlusion.
Experimental results showed that the MDYOLO framework outperformed other state-of-the-art license plate detection methods, including the original YOLO framework, in terms of accuracy and robustness to poor lighting conditions and occlusions. The authors attributed this improvement to the ability of the model to capture multi-directional license plates and to the use of a novel loss function that penalized the model for false positives and false negatives.
Overall, the MDYOLO framework offers a promising approach to license plate detection, particularly in scenarios where multi-directional license plates are common and lighting conditions may be poor. The figure 2.2 below shows the redesigned CNN architecture for the MDYOLO model.

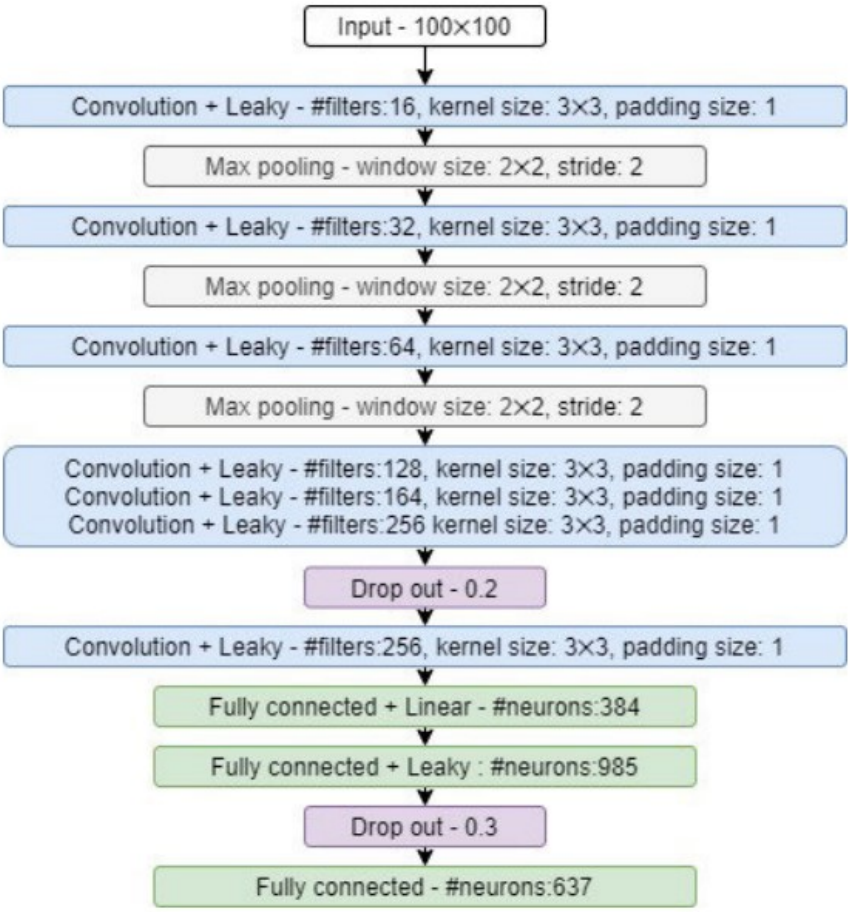

Figure 2.2: CNN architecture for the MD-YOLO model
[58]

2.4 Character Segmentation

2.4.1 Character Segmentation Using Pixel Connectivity

In [31] and [56] used Pixel connectivity , It is a method commonly used for character segmentation in license plate recognition. It involves labeling connected pixels in the license plate image and extracting character regions based on predetermined size or aspect ratio criteria. This approach is relatively simple to implement and is robust against rotated license plates. However, pixel connectivity-based methods may struggle with broken characters or joined characters resulting from binarization threshold issues. Nonetheless, they offer a streamlined solution by eliminating the need for extensive pre-processing steps to handle license plate rotation, making them an attractive choice in license plate recognition systems.

2.4.2 Character segmentation using projection profiles

Projection profile methods take advantage of the contrast between character and background pixels in the binarized license plate image. Typically, vertical projections are used to identify the starting and ending positions of each character, while horizontal projections are used to extract the individual characters. This approach is not affected by the specific positions of the

characters, making it robust and independent of character arrangement. However, projection-based techniques are sensitive to image quality and noise, which can negatively impact their performance. To mitigate this, a denoising stage is often included in the pre-processing phase of the recognition pipeline. While projection-based methods may have lower robustness compared to pixel connectivity-based methods for rotations, they still offer good rotation robustness and can effectively handle license plates with varying character positions.[13]

2.4.3 Character segmentation using deep learning

Character segmentation using deep learning has gained popularity as a modern approach in license plate recognition. It involves utilizing convolutional neural networks (CNNs) for computer vision tasks [20]. The process entails feeding a localized license plate image into the CNN, which then generates bounding boxes around individual characters as its output. Although deep learning-based segmentation achieves accurate results, it typically requires more time and computational resources compared to traditional computer vision techniques, depending on the dataset. However, some recent license plate recognition pipelines using deep learning have opted for implicit character segmentation in later stages, reducing the number of parameters and computational costs involved [27], [21].

2.5 License plate recognition

2.5.1 Convolutional Neural Network

In the study referenced in [3], Alam proposed a method for identifying and recognizing Bengali language vehicle number plates using Convolutional Neural Network (CNN) and Deep Learning strategies. The main idea behind this method was to develop a system that can accurately recognize and store vehicle numbers in a cloud-based system. To achieve this goal, the proposed system utilized a super-resolution technique with CNN in the recognition portion to reconstruct the pixel quality of the input image. This technique helped to improve the image resolution and overall accuracy of the recognition process.

The number plate recognition process involved segmenting each character of the number plate using a bounding box technique. This allowed the system to accurately identify each character and recognize the entire number plate.
To test the effectiveness of the proposed system, Alam used 700 vehicles and evaluated the outcomes on a validation set and an evaluation set. The results showed that the CNN achieved 98.2% accuracy in the validation set and 98.1% accuracy in the evaluation set, with an error rate of 1.8%.[3]
In the study referenced in [42], Shaifur Rahman proposed a Bangla license plate recognition system (BLPRS) based on Convolutional Neural Networks (CNN). The primary aim of this system was to develop an efficient and accurate method for recognizing Bangla license plates that could be used for various purposes, such as roadside assistance and vehicle license status identification.
To achieve this goal, the authors used six CNN layers and a fully connected layer for training the BLPRS. These layers were designed to extract and analyze the relevant features of the license plate images and recognize the license plate numbers accurately. To test the effectiveness of the proposed BLPRS, the authors used a testing dataset and reported that they achieved 89% testing precision, indicating that the system was highly accurate in recognizing Bangla license

plates.[42]

In their work, Liu et al. [24] proposed a method for license plate recognition that combined connected component analysis and projection analysis for segmentation, as well as two CNNs for character recognition. The proposed method was capable of handling overexposed license plates by first converting them to grayscale and enhancing the contrast ratio of images. Then, breadth-first search algorithm was used to obtain the connected components and determine whether there were any missing or redundant characters.
Two CNNs were designed for character recognition, one for Chinese characters and the other for recognizing numbers and letters. The SCNN and RCNN were designed to be simple and recurrent, and were trained on a dataset of 2189 images. The authors reported a segmentation rate of 96.58% and a recognition rate of 98.09%. The proposed method outperformed previous methods in terms of accuracy and robustness to overexposed license plates.[24]

2.5.2 Recurrent Neural Network (RNN)

Li et al in [22]. proposed a license plate recognition system based on a Bidirectional Recurrent Neural Network (BRNN) with Connectionist Temporal Classification (CTC) loss. The system is designed to process the license plate features extracted by the Region Proposal Network (RPN) and recognize the license plate characters. The BRNN architecture consists of two recurrent neural networks (RNNs), one that processes the input sequence in the forward direction and another that processes it in the backward direction. The outputs of these two RNNs are combined to generate the final output. The use of BRNNs allows the system to capture both past and future contextual information in the input sequence, which is important for accurate character recognition.
The CTC loss function is used to train the BRNN model. It allows the model to learn to recognize sequences of characters without the need for explicit alignment between the input sequence and the output sequence. This is important for license plate recognition, where the number of characters in a license plate may vary and the characters may be arranged in different ways.
Overall, the proposed system achieved high accuracy in character recognition on various datasets, demonstrating the effectiveness of the BRNN with CTC loss for license plate recognition. Wang, Yi, et al. "Rethinking and designing a high-performing automatic license plate recognition approach." IEEE Transactions on Intelligent Transportation Systems 23.7 (2021): 8868-8880.[22]
The study referenced in [23] proposed a method for license plate recognition based on recurrent neural networks (RNNs) with Long Short-Term Memory (LSTM) to recognize sequential features extracted from the license plate images via convolutional neural networks (CNNs). In this approach, each detected license plate image was first converted to a grayscale image and resized to 24x94 pixels. Then, a sliding window technique was used to partition the image into sub-windows of 24x24 pixels with a step size of 1. Each partitioned image patch was fed into a 36-class CNN classifier to extract sequence features.
The fourth convolutional layer and the first fully connected layer were concatenated together into one feature vector with a length of 5096. Principal Component Analysis (PCA) was used to reduce the feature dimension to 256 dimensions, followed by feature normalization. Finally, Connectionist Temporal Classification (CTC) was designed to decode the predicted probability sequence into output labels directly, with an average recognition rate of around 92.47%.
The proposed method demonstrated significant improvements in license plate recognition accuracy compared to previous methods. The use of RNNs with LSTM allowed for better recog-

nition of sequential features in the license plate images, leading to higher accuracy in license plate recognition. The approach showed promising results for real-world applications such as traffic monitoring, toll collection, and parking management.[23]

2.5.3 Optical Character Recognition (OCR)

In the study referenced in [11], Kashyap introduced an automated number plate recognition (ANPR) system that uses image processing techniques to recognize license plate characters. The system was designed to automatically capture and analyze license plate images to extract the necessary information, such as the license plate number. To recognize the characters on the license plate, the system used Optical Character Recognition (OCR) technology. OCR is a process that converts the lettering on an image to text, allowing the system to read and recognize the license plate number.
To evaluate the performance of the proposed ANPR system, Kashyap used a testing dataset and reported that they achieved an accuracy of about 82.6%. While this level of accuracy is relatively high, it still leaves room for improvement, as it may not be sufficient for certain applications, such as law enforcement or toll collection.[11]

In the study referenced in [19], Pechiammal proposed an effective process for automatic license plate recognition (ALPR). The proposed method consisted of three portions: segmentation of characters, identification of optical characters, and matching of models. The primary objective of the proposed method was to extract the characters from the license plate accurately and efficiently.
The first portion of the proposed method was the segmentation of characters, which involved identifying the characters on the license plate image and separating them from the background. This step was crucial, as it allowed the system to focus only on the characters and ignore any irrelevant information.
The second portion of the proposed method was the identification of optical characters. In this step, Optical Character Recognition (OCR) technology was used to recognize the characters on the license plate and convert them into machine-readable text.
The final portion of the proposed method was the matching of models. In this step, the system compared the recognized characters with a pre-defined database of license plate characters to identify the license plate number accurately.
To evaluate the effectiveness of the proposed method, Pechiammal used a testing dataset and reported that they achieved an 85% extraction rate. This means that the proposed method was able to extract the license plate characters accurately in 85% of the cases.[19]

In the study referenced in [39], Rehman proposed an innovative vehicle number plate recognition method using Optical Character Recognition (OCR) and template matching strategies for the Pakistani language. The proposed method was evaluated on several real-time images of different formats of number plates used in Pakistan.
The primary objective of the proposed method was to develop an Automatic Number Plate Recognition (ANPR) system that could be used by law enforcement agencies and private organizations to enhance home security, while also providing time and money-saving benefits.
To achieve this objective, the proposed ANPR system used OCR technology to recognize the characters on the license plate and convert them into machine-readable text. The system also used template matching strategies to compare the recognized characters with pre-defined database of license plate characters to identify the license plate number accurately. 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 cases.[39]

2.5.4 K-means Clustering-based Approach

In [35], Pustokhina proposed an efficient deep learning-based approach for license plate recognition for vehicles. The proposed approach consists of two main stages: optimal K-means clustering segmentation and Convolutional Neural Network (CNN) based character recognition.
The proposed method first uses the Bernsen Algorithms (IBA) and the Connected Component Analysis (CCA) models to classify and locate the license plates. Then, the optimal K-means clustering algorithm is used to segment the license plate. Finally, a Convolutional Neural Network (CNN) is employed to recognize the characters of the segmented license plate. The proposed method was evaluated on different datasets, and the results showed that the maximum accuracy obtained by the proposed Optimal K-Means with Convolutional Neural Network (OKM-CNN) system on the datasets is about 98.1%. The proposed method is efficient and accurate, and it can be used for real-time applications such as traffic monitoring, parking management, and toll collection systems.[35]

2.5.5 Template matching

In [46], the Template Matching technique was used to recognize the characters segmented from Moroccan format number plates. This method involves matching the segmented characters with a pre-defined template of characters to identify the characters accurately. The proposed system was evaluated on four different sets of Moroccan format number plates, and the success rates achieved were 98.1%, 96.37%, 93.07%, and 92.52%, respectively. These results indicate the effectiveness of the Template Matching technique in recognizing characters from Moroccan format number plates. However, it is worth noting that this method may not be as robust as other deep learning-based methods and may have limitations in handling variations in license plate formats and lighting conditions.[46]

2.6 Conclusion

In conclusion, there are various methods for license plate detection and recognition, each with its strengths and weaknesses. Deep learning-based approaches, particularly those using convolutional neural networks (CNNs), have shown promising results in recent years, achieving high accuracy rates in both detection and recognition tasks. Other techniques such as template matching, Recurrent neural network (RNN), and optical character recognition (OCR) have also been used in combination with deep learning approaches to further improve performance. Overall, the choice of method depends on the specific application, the type of license plate, and the available resources.

Chapter 3

Conception

3.1 Introduction

In this chapter, we propose an approach for license plate detection and recognition using deep learning techniques. Our system combines the power of the YOLO algorithm for license plate detection and a CNN model for character recognition. The goal is to efficiently detect license plates and accurately recognize the characters on them. By leveraging deep learning, we overcome challenges such as varying license plate styles and lighting conditions. This chapter presents the objectives of our system, the architecture, the design steps, and the proposed algorithms. We conclude with insights into the potential applications of our approach.

3.2 System goals

The primary goal of this project is to create a deep learning system capable of detecting and extracting license plate images from a given input, as well as accurately recognizing and extracting the content of the license plates. By leveraging advanced deep learning techniques, the system aims to identify and extract the necessary information, specifically the vehicle number, from the license plate images.

3.3 Characteristics of Algerian plates

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

Figure 3.1: Algerian license plate.

The registration mark of the Algerian license plate follows a specific format consisting of four groups of digits. It is important to note that these groups are read from right to leftas follow:

- 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 first 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 Architecture of the system

The architecture diagram below illustrates how this comprehensive workflow enables accurate identification and classification of license plate characters from image inputs. The proposed system consists of the following phases:

- License Plate Detection: This phase detects the license plate from the image, even in challenging conditions such as skew problems and unbalanced lighting.
- License Plate Extraction: This phase completes the previous phase by extracting and segmenting the license plate from the detected image.
- License Plate Preprocessing: In this phase, necessary filtering, correction, and enhancement techniques are applied to prepare the license plate image for the next steps.
- Character Segmentation and Character Recognition: At this stage, the enhanced and extracted license plate image is used for character box extraction, followed by digitization of the extracted boxes.
- Character Classification: The final phase involves classifying the digitized character boxes into numbers.

This systematic approach ensures the accurate identification and classification of license plate characters, even in various challenging scenarios.

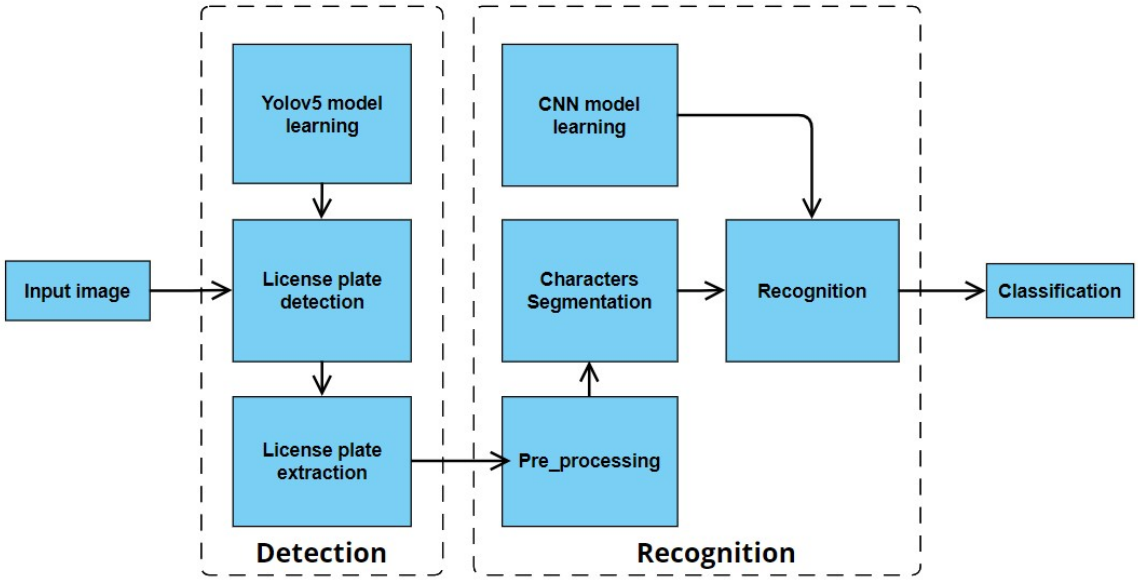

Figure 3.2: The architecture of the proposed license plate recognition system.

3.5 License plate detection

3.5.1 Yolov5 model learning

YOLOv5 is the latest iteration of the YOLO object detection algorithm, introduced in 2020. It offers four versions: YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, with increasing depth and feature map width. The network structure consists of four components: input, backbone, neck, and prediction. During input processing, data augmentation and anchor frame calculation are performed, enhancing detection capabilities. The backbone incorporates the Focus and CSP structures, while the neck combines FPN and PAN structures for feature fusion. The prediction stage utilizes the CIOU loss function and DIOU_nms operation to improve accuracy. These design choices contribute to the efficiency, precision, and lightweight nature of YOLOv5.[44] Figure 3.3 illustrates the architectural overview of the main steps involved in the YOLOv5 algorithm.

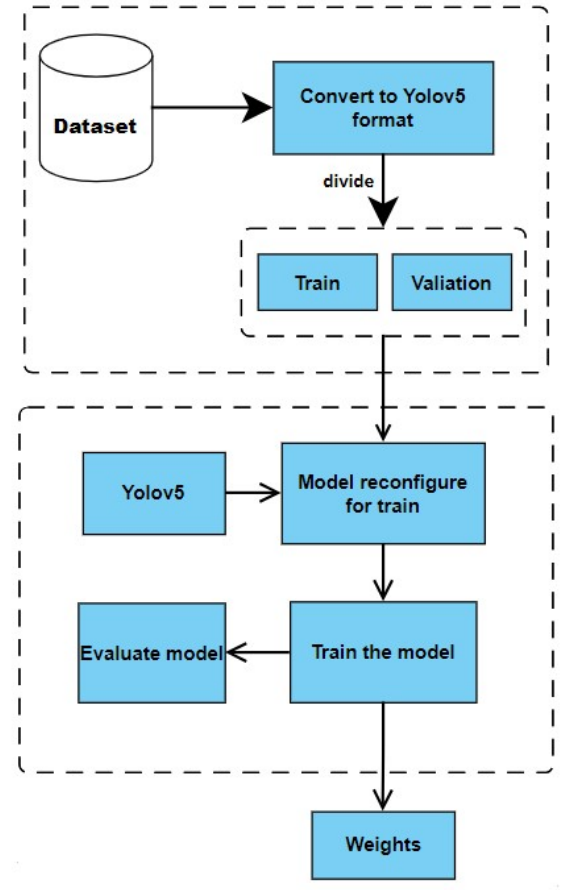

Figure 3.3: The Basic Phases of YOLOv5 Model.

Prepare the dataset The process of dataset selection posed a significant challenge due to the unavailability of a pre-existing dataset specifically focused on Algerian vehicle plates. To overcome this obstacle, a multi-faceted approach was adopted. Initially, a series of images capturing Algerian vehicle plates were obtained using a camera phone. Additionally, images featuring Algerian vehicle plates were collected from online platforms such as Ouedkniss [1] . In order to enhance the dataset's diversity and expand its scope, an additional dataset from Kaggle [4] , comprising 432 annotated images of vehicle plates from various countries, was

incorporated. By combining these datasets. The dataset collected was meticulously annotated to provide accurate and detailed information about the objects of interest. Following the annotation process, the dataset was divided into two distinct folders: a training folder, and a validation folder. This division ensured a well-balanced distribution of data for training, and evaluation purposes.
Subsequently, the annotations for each image were converted from their original XML format to a YOLO readable format. This conversion involved transforming the annotations into text files, with each file corresponding to an individual image and stored in a designated "labels" folder.
To streamline the training process, the number of classes within the dataset was standardized to a single class, specifically labeled as "license_plate ". This simplification allowed for a focused and efficient model development, where the primary objective was the accurate detection and classification of license plates.

Download YOLO The first step it was download yolov5 [50], Next, navigate to the yolov5/data folder and create license_plate.yaml file.
The license_plate.yaml file is a configuration file used for training an object detection model with YOLOv5. It specifies the paths to the training, and validation images directories. Additionally, it defines that there is one class to be detected, which is "license_plate". The file provides the necessary information for setting up the training process, including the class names and the number of classes.

Train model The training process for the license plate detection model is initiated using the YOLOv5 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 32, which means that during each iteration of training, the model processes 32 images together. This helps optimize the training process.
The training is carried out for 60 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 license plate detection task.
The license_plate.yaml file is utilized to provide the necessary configuration for the training process.

Model Evaluation To assess the effectiveness of the trained license plate detection model, an evaluation process is conducted using the YOLOv5 architecture. The command initiates the evaluation by executing the val.py script. This script is responsible for evaluating the model's performance in accurately detecting license plates. The trained model's weights, specifically the best-performing ones, are provided through the –weights parameter, pointing to the best.pt file. The configuration of the evaluation is specified by the license_plate.yaml file, which contains essential details about the dataset and class information. During the evaluation, images are resized to 640x640 pixels using the –img parameter to maintain consistency with the training process. The –task test argument indicates that the evaluation is conducted on the evaluation dataset, which ensures a reliable assessment of the model's ability to generalize to unseen license plate images. By executing this command, the val.py script evaluates the trained model's performance, providing valuable insights into its accuracy and its potential applicability in practical scenarios.

3.5.2 License plate detection

In this step, the YOLOv5 model is utilized to detect the license plate from the input image. The following is a more formal explanation of the process:

1. Load the YOLOv5 model: The pre-trained YOLOv5 model is loaded using the torch.hub.load() function from the ultralytics/yolov5 repository [50]. The path to the trained weights file is provided through the path parameter. Setting force_reload=True ensures that the latest version of the model is loaded.

2. The loaded YOLOv5 model is used to perform inference on the input image by passing it as an argument to the model() function. The results of the inference, containing the bounding box coordinates and class labels, are stored in the results variable.

3.5.3 Extract the license plate

The license plate region is extracted by utilizing the image with detection, which provides the necessary bounding box coordinates. The YOLOv5 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.

3.6 License plate recognition

3.6.1 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:

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.

Grayscale The resized image is converted from the BGR color space to grayscale . This conversion simplifies subsequent image processing steps.Figure 3.4 displays an example of a gray-scale license plate.

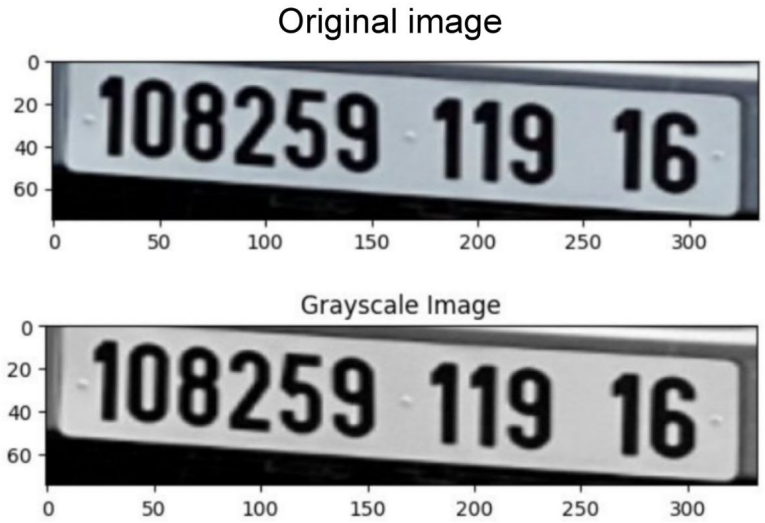

Figure 3.4: Example Of Gray-Scale License Plate.

Histogram Equalization Histogram equalization is a contrast enhancement technique that redistributes the pixel intensities of an image to improve its contrast. The algorithm for histogram equalization is presented below:

Let $H(i)$ denote the histogram of the input image, where i ranges from 0 to $L-1$, and L is the number of intensity levels.

Calculate the cumulative distribution function (CDF) by summing up the histogram values:

$$CDF(i) = \sum_{j=0}^{i} H(j), \text{ for } j = 0 \text{ to } i. \tag{3.1}$$

Normalize the CDF values by dividing each CDF value by the total number of pixels in the image, denoted as N:

$$CDF_{\text{norm}}(i) = \frac{CDF(i)}{N}. \tag{3.2}$$

Map the intensity values using the normalized CDF values:

$$M(i) = \text{round}(CDF_{\text{norm}}(i) \cdot (L-1)). \tag{3.3}$$

Create an equalized image by assigning the mapped intensity values to the corresponding pixels.

By applying this algorithm, the intensities of the image are spread evenly across the entire range, resulting in enhanced contrast and improved visibility of details in the image.[7] Figure 3.5 showcases an example of image light enhancement.

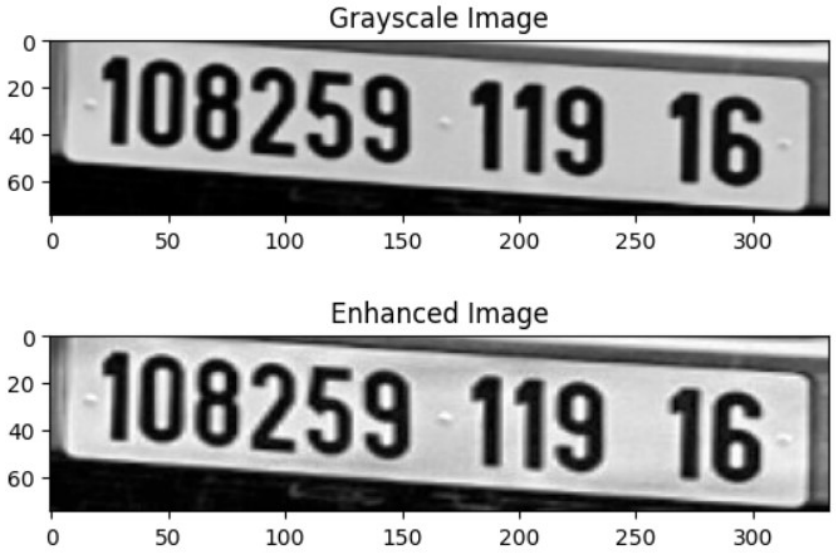

Figure 3.5: Example Enhanced License Plate.

Gaussian blur Gaussian blur is an image filtering technique that applies a blur effect by convolving the image with a Gaussian kernel. It is particularly useful for enhancing the character segmentation phase, which in turn improves the accuracy of subsequent phases in the license plate recognition system. Mathematically, the convolution operation between the input image (I) and the Gaussian kernel (G) can be represented as follows:

$$O(x,y) = \sum_a \sum_b [I(a,b) \cdot G(x-a, y-b)], \tag{3.4}$$

where (a,b) represents the kernel coordinates.[53] Here is an example of applying gaussian blur on license plate:

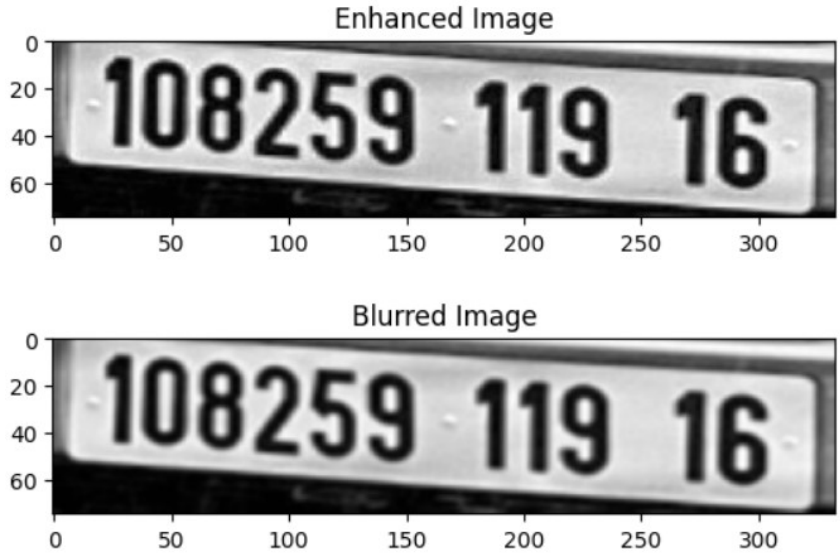

Figure 3.6: Example of blurred image.

Otsu's thresholding Otsu's binarization calculates the threshold by considering the variance σ within each class (foreground and background) and aims to minimize the variance within

each class while maximizing the variance between the classes. The formula for finding the within-class variance at any threshold t is given by:

$$\sigma^2(t) = \omega_{bg}(t) \cdot \sigma^2_{bg}(t) + \omega_{fg}(t) \cdot \sigma^2_{fg}(t) \tag{3.5}$$

Where $\omega_{bg}(t)$ and $\omega_{fg}(t)$ represent the probabilities of the number of pixels. This threshold is essential for preparing the license plate image to be segmented into characters. By determining the optimal threshold value using Otsu's thresholding technique, the license plate image can be effectively separated into foreground (characters) and background regions. This segmentation step is crucial for further processing and accurately recognizing the characters on the license plate.[32] Here is an examle in Figure 3.7

Figure 3.7: Example of Otsu's thresholding license plate .

Morphological operations Morphological operations [37] are image processing techniques used to modify the shape and structure of objects in an image (I). They involve simple operations such as dilation (D) and erosion (E). These operations are performed using a small matrix called a structuring element (SE) that slides over the image, defined as follows:

$$E(I, SE) = \{x \mid SE \text{ is a subset of } I, \text{ centered at } x\} \tag{3.6}$$

$$D(I, SE) = \{x \mid \exists y \text{ in } SE \text{ such that } I(x-y) = 1\} \tag{3.7}$$

We utilized these two morphological operations, erosion and dilation, to enhance the character extraction process and eliminate small regions that could potentially mislead the results. By applying erosion, we could shrink the boundaries of the characters, helping to separate them from the background and reduce any noise or unwanted artifacts. On the other hand, dilation expanded the character boundaries, filling in any gaps and ensuring the characters were well-connected. By iteratively applying these operations with Square structuring element, we could refine the character regions and improve the accuracy of subsequent recognition and classification steps in the license plate recognition system. Here are two examples in Figures 3.8 and 3.9.

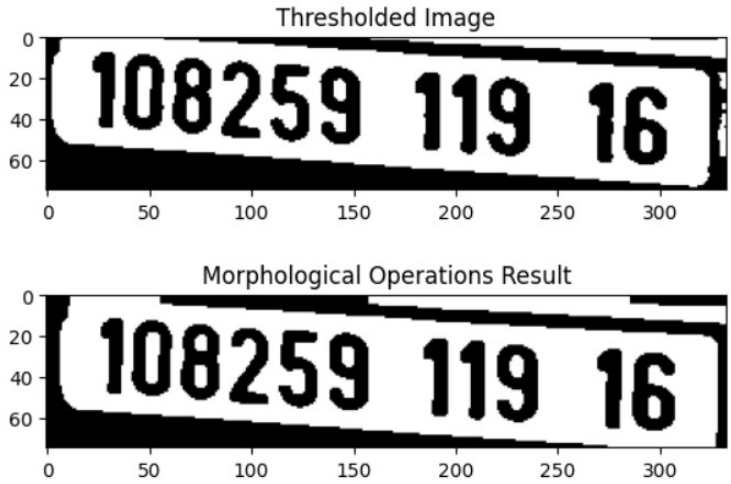

Figure 3.8: Example 1 of morpholoical operation.

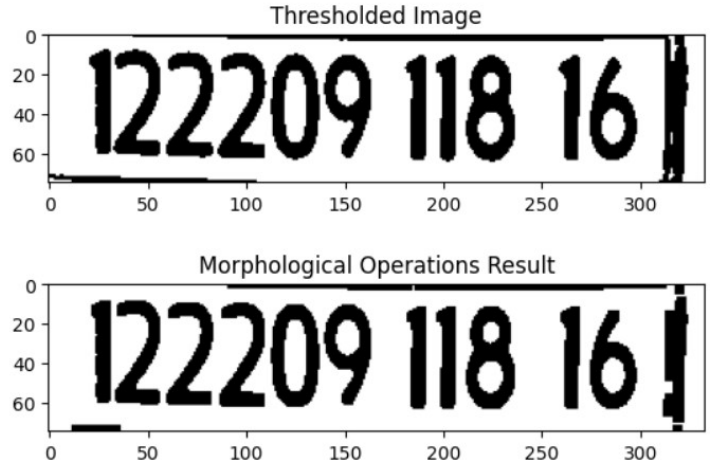

Figure 3.9: Example 2 of morpholoical operation.

Straighten the license plate This method aims to correct any skew that may be present in the license plate image. The following steps are involved in the skew correction process:

- Edge Detection: The first step is to perform edge detection on the license plate region of the image. This helps in identifying the edges or boundaries of objects within the plate.
- Hough Transform: The detected edges are then passed through the Hough transform algorithm. This algorithm identifies lines in the image by looking for patterns of points that form a line. In this case, the cv2.HoughLinesP function is used, which returns a list of line segments
- Angle Calculation: For each line segment detected, the angle is calculated using the coordinates of the start and end points. This is done by applying the inverse tangent function (np.arctan2) to the difference in y-coordinates and x-coordinates. The resulting angle is converted from radians to degrees.

- Median Angle: The list of calculated angles is then used to find the median angle. The median angle helps to estimate the overall skew of the license plate.
- Rotation: With the median angle determined, a rotation matrix (cv2.getRotationMatrix2D) is created. The rotation matrix specifies the transformation required to rotate the image by the median angle around a specific center point. The center point is typically set as the center of the image.
- Warped Affine Transformation: The rotation matrix is applied to the license plate region of the image using the cv2.warpAffine function. This function performs an affine transformation on the image, applying the rotation specified by the rotation matrix. The resulting image is straightened or aligned based on the estimated skew.As shown in the example in Figure 3.10.

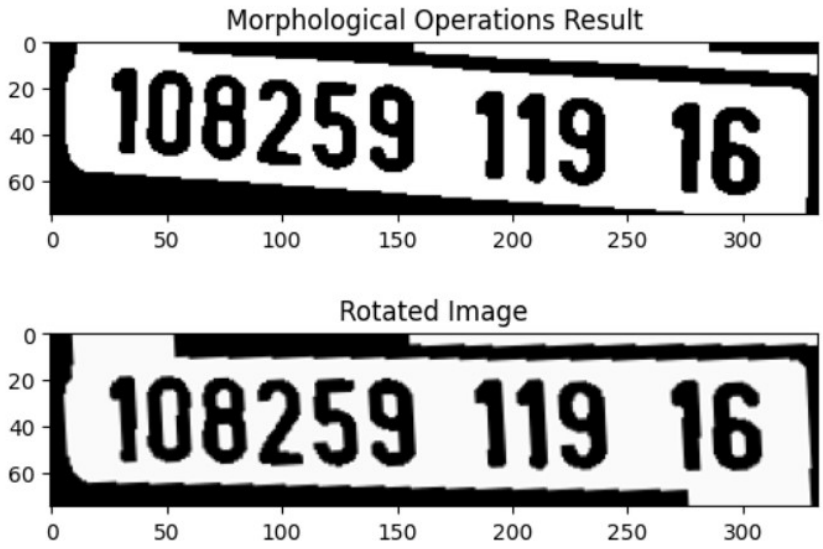

Figure 3.10: Example of skew correction.

3.6.2 Character segmentation

We utilized a specialized function designed specifically for segmenting and extracting individual characters from a given license plate image. This function takes the pre-processed license plate image as input and executes a sequence of operations to identify and extract the characters.

Initially, the function defines the ‘find_contours(dimensions, img)‘ nested function, which utilizes OpenCV’s contour detection algorithm to identify contours within the image. The contours are then sorted based on their area in descending order, with the intention of selecting the largest contours that correspond to the license plate and characters.

The function iterates through the sorted contours and checks their dimensions against predefined lower and upper bounds. This filtering step helps eliminate noise and select contours that are likely to represent characters. The function extracts each character by defining a bounding rectangle around the contour and resizing it to a standard size of 20x40 pixels.

To visualize the segmentation process, the function draws rectangles around the identified characters on the license plate image. The modified image is saved as ‘predict.jpg‘.

The characters are further processed to prepare them for classification. The function inverts the colors of the characters and resizes them to a final size of 24x44 pixels, with a black border. The processed character images are stored in a list called ‘img_res‘.

To ensure the correct order of the characters, the function sorts the contour indices based on their x-coordinates. The character images are then rearranged according to the sorted indices. The final sorted character images are stored in the ‘img_res‘ list.

Lastly, the function returns the sorted character images as a NumPy array.

3.6.3 Convolutional Neural Network (CNN) learning model

Prepare the dataset We have obtained a character recognition dataset from GitHub for our project.[15] GitHub is a widely used platform for hosting and sharing code repositories, including datasets. By accessing a character recognition dataset from GitHub, we aim to leverage the existing data to train and develop our own character recognition models or perform various analyses.

Explain the model A convolutional neural network (CNN) model is proposed for image classification. The model is implemented using TensorFlow’s Keras [48] module. The CNN architecture consists of several layers designed to extract meaningful features from input images and make accurate predictions.

The first layer is a 2D convolutional layer with 32 filters of size 5x5. It applies these filters to the input images and uses the Rectified Linear Unit (ReLU) activation function to introduce non-linearity. The output feature maps have the same spatial dimensions as the input images due to the "same" padding used.

Next, a max pooling layer with a pool size of 2x2 is applied. This layer reduces the spatial dimensions of the feature maps, capturing the most important features while reducing computational complexity.

To prevent overfitting, a dropout layer with a dropout rate of 0.4 is included. This layer randomly sets a fraction of the input units to zero during training, increasing the model’s robustness.

The flattened feature maps are then passed to a fully connected layer with 128 units and a ReLU activation function. This layer learns complex patterns and relationships from the flattened feature vectors.

Finally, a dense output layer with 10 units is added. It uses the softmax activation function to generate class probabilities for multi-class classification. The model is trained to minimize the categorical cross-entropy loss and maximize the accuracy of the predictions.

The model is compiled with the Adam optimizer, which adjusts the learning rate during training, and a learning rate of 0.00001 is set. The metrics used to evaluate the model's performance during training are accuracy.

Train the model We trained the model using 'fit()' function by iteratively processing batches of training data over a specified number of epochs. In our case, the batch size is set to 20, meaning that the model processes one sample at a time. During training, the model adjusts its parameters to minimize the loss function using the specified optimizer. The validation data is used to evaluate the model's performance after each epoch. By training the model for 200 epochs, this step aims to optimize the model's performance by iteratively adjusting its parameters based on the training data and monitoring its performance on the validation data.

Evaluate model The evaluation of the model on the validation data is an important step in assessing its performance. By using the 'evaluate()' function with the validation data generator, the model's loss and accuracy on the validation set are calculated. The validation loss indicates the level of agreement between the model's predictions and the actual labels, with lower values indicating better performance. The validation accuracy represents the percentage of correctly classified samples. These metrics provide valuable insights into the model's effectiveness in generalizing to unseen data. In the context of my dissertation, this evaluation step is crucial for evaluating the model's performance and validating its ability to accurately classify the target classes.

3.6.4 Character recognition

License plate number extraction using predictive image analysis involves iterating over each character in the predicted images. The character images are prepared and processed for clarity. the pre_trained model of CNN is utilized to predict the class probabilities for each character. The character with the highest probability is selected, resulting in the extraction of the text number of the license plate. This procedure enables accurate and efficient recognition of license plate numbers .

3.6.5 Classification

In this step the process involves reading license plate numbers from a file. It then proceeds to match the extracted characters from the license plate with corresponding values in different database files. To obtain the final result which is the classification of the number plate recognized before.As shown in Figure 3.11.

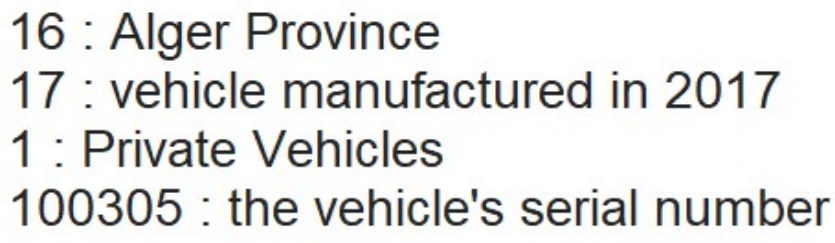

Figure 3.11: Example of classification.

3.7 Conclusion

In conclusion, the conception chapter has presented the global architecture that forms the basis of our project. It has outlined the fundamental steps involved in our work, providing a clear overview of the process. The next chapter will delve into the implementation details, where we will translate the architectural framework into practical solutions.

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 a pc which has the following specifications :

Model Part	Used Laptop
Processor	Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50 GHz
RAM	8.00 GB
System type	64-bit operating system, x64-based processor
Edition	Windows 10 Pro

Table 4.1: Characteristics of the material used.

4.3 Programming language

4.3.1 Python

Python is described as a high-level programming language that emphasizes code readability and simplicity. It is known for its clear and concise syntax, which allows programmers to express concepts in fewer lines of code compared to other programming languages. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming styles. It has a vast standard library and a large ecosystem of third-party packages, making it suitable for a wide range of applications. Python is widely used in various fields such as web development, scientific computing, data analysis, artificial intelligence, and automation. It is an open-source language, freely available for use and distribution, with an active community contributing to its development and improvement.[36]

4.3.2 Jupyter notebook

Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and explanatory text. It supports various programming languages, including Python, R, and Julia. Jupyter Notebook provides an interactive computational environment where you can write and execute code in a structured and organized manner. It has gained popularity among data scientists, researchers, and educators for its ability to combine code, documentation, and visualizations into a single, easily shareable document.[17]

4.4 Libraries

4.4.1 Tensorflow

TensorFlow is an open-source machine learning framework developed by Google. It provides a comprehensive ecosystem of tools, libraries, and resources for building and deploying machine learning models. TensorFlow supports both deep learning and traditional machine learning algorithms and offers high-level APIs for easy model development, as well as lower-level APIs for greater flexibility and customization. It is widely used in various domains, including computer vision, natural language processing, speech recognition, and more.[16]

4.4.2 Keras

Keras is a user-friendly, high-level neural networks API in Python. It provides a simple and efficient way to build and experiment with deep learning models. With Keras, developers can easily define and configure neural networks using intuitive building blocks, making it accessible to both beginners and experienced researchers. It supports multiple backends and offers tools for fast experimentation, making it a popular choice for deep learning tasks.[18]

4.4.3 PyTorch

PyTorch is a widely used open-source machine learning framework and scientific computing library. It offers a flexible and intuitive approach to building and training deep learning models. With its dynamic computation graph and seamless integration with NumPy, PyTorch simplifies the process of manipulating tensors and performing mathematical operations. It prioritizes GPU acceleration, enabling faster training and inference, and supports parallel computing and distributed training. PyTorch's rich ecosystem includes specialized libraries for computer vision, natural language processing, and audio processing, along with pre-trained models and extensive learning resources. Its popularity stems from its user-friendly interface and powerful capabilities, making it a preferred choice for researchers and practitioners in the deep learning community.[28]

4.4.4 OpenCv

OpenCV (Open Source Computer Vision) is a versatile and widely used open-source library for computer vision tasks. It offers an extensive collection of functions and algorithms for tasks like image and video processing, object detection, and feature extraction. OpenCV is known

for its cross-platform compatibility and is available in multiple programming languages, including Python. With its user-friendly API and comprehensive documentation, OpenCV enables developers to integrate computer vision capabilities into their projects seamlessly. Its broad range of applications, from robotics to medical imaging, coupled with its active community and pre-trained models, makes OpenCV a popular choice for computer vision enthusiasts and professionals alike.[8]

4.4.5 Matplotlib

Matplotlib is a powerful data visualization library for Python. It provides a wide range of plotting functions and customization options to create visually appealing and informative plots. With Matplotlib, users can easily generate line plots, scatter plots, bar plots, histograms, and more. It offers precise control over plot elements, such as colors, labels, and axes, allowing for highly customizable visualizations. Matplotlib is compatible with other Python libraries, making it a popular choice for data analysis and scientific visualization. Its versatility, user-friendly interface, and extensive documentation make it an essential tool for visualizing data in a concise and effective manner.[49]

4.4.6 NumPy

NumPy (Numerical Python) is a fundamental library for scientific computing in Python. It provides efficient and powerful tools for working with multi-dimensional arrays and performing mathematical operations on them. With NumPy, users can easily manipulate and analyze data, perform linear algebra operations, generate random numbers, and more. Its integration with other scientific libraries makes it a popular choice for tasks such as data analysis, machine learning, and simulation. NumPy's simplicity, speed, and versatility make it an essential tool for numerical computing in Python.[9]

4.4.7 scikit-image (skimage)

skimage is a Python library specifically designed for image processing and computer vision tasks. It offers a wide range of algorithms and functions to manipulate, analyze, and enhance images efficiently. With skimage, users can perform various operations such as filtering, segmentation, and feature extraction. The library also provides tools for image registration, geometric transformations, and object detection. skimage is known for its user-friendly interface, well-documented API, and seamless integration with other scientific Python libraries. Its extensive capabilities and ease of use make it a popular choice for image processing tasks in a wide range of applications.[52]

4.4.8 Tkinter

Tkinter is a Python library that enables the creation of graphical user interfaces (GUI) for desktop applications. It provides a range of tools and widgets to design interactive and visually appealing interfaces. Tkinter is based on the Tk GUI toolkit and offers a straightforward and intuitive approach to building GUI applications. With Tkinter, developers can create windows, buttons, menus, text boxes, and other GUI elements, and organize them using layout managers. The library also includes event handling mechanisms to capture user interactions. Tkinter is

widely used due to its simplicity, cross-platform compatibility, and ability to develop GUI applications in Python with ease.[40]

4.4.9 Customtkinter

"Customtkinter" refers to a customized version of the Tkinter library in Python. It involves making modifications or extensions to the standard Tkinter library to cater to specific requirements or add new features. With "customtkinter," developers have the flexibility to introduce unique widgets, alter the appearance or behavior of existing widgets, or create custom layout managers. These customizations allow for more tailored and specialized user interfaces in Python applications. The specific features and enhancements provided by "customtkinter" may vary depending on the implementation and customization choices made by the developer.[40]

4.4.10 PIL (Python Imaging Library)

PIL is a popular Python library for image processing tasks. It provides a wide range of functionalities to open, manipulate, and save different image file formats. PIL offers capabilities for resizing, cropping, rotating, and filtering images, as well as adjusting their colors, contrast, and brightness. The library also supports basic image editing operations like adding text, drawing shapes, and applying various image effects. PIL is known for its simplicity and ease of use, making it a versatile tool for working with images in Python. It has been widely adopted and is often used in applications related to computer vision, graphics, and multimedia processing.[33]

4.5 System overview

We dedicate this part to showing the interface and function of our system, specifically focusing on the main modules. The figure below illustrates the home interface of our application.

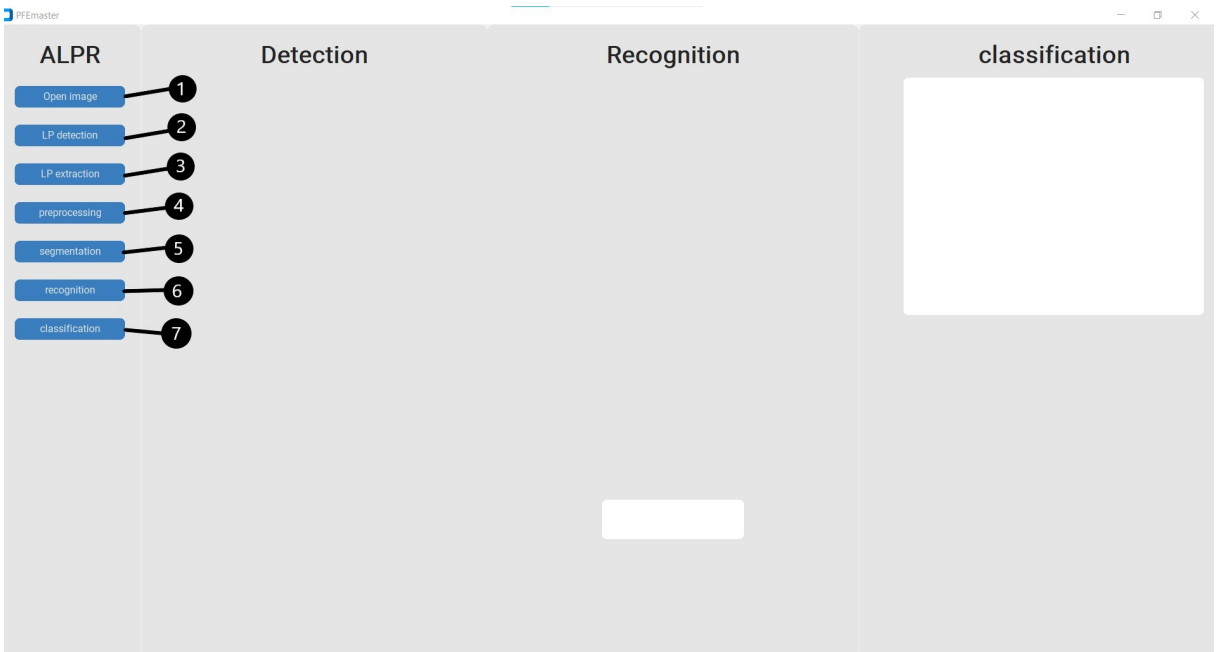

Figure 4.1: The home interface of our system.

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

1. The initial button enables the user to open an image .

2. The second button is designed to perform the task of license plate detection and subsequently draw a bounding box around the identified license plate.

3. The third button is responsible for extracting the detected license plate from the image.

4. The fourth button is responsible for performing preprocessing operations on the detected license plate in order to facilitate its recognition.

5. The fifth button is utilized for the purpose of segmenting the characters of the license plate.

6. The sixth button is responsible for recognizing the segmented characters of the license plate.

7. The seventh button is dedicated to classifying or categorizing the extracted number from the license plate.

4.6 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 license plate recognition :

1. To commence the license plate detection process, we start by opening an image file using the command "open image." This allows us to access the desired image and utilize it for further analysis and detection procedures.As shown in Figure 4.2.

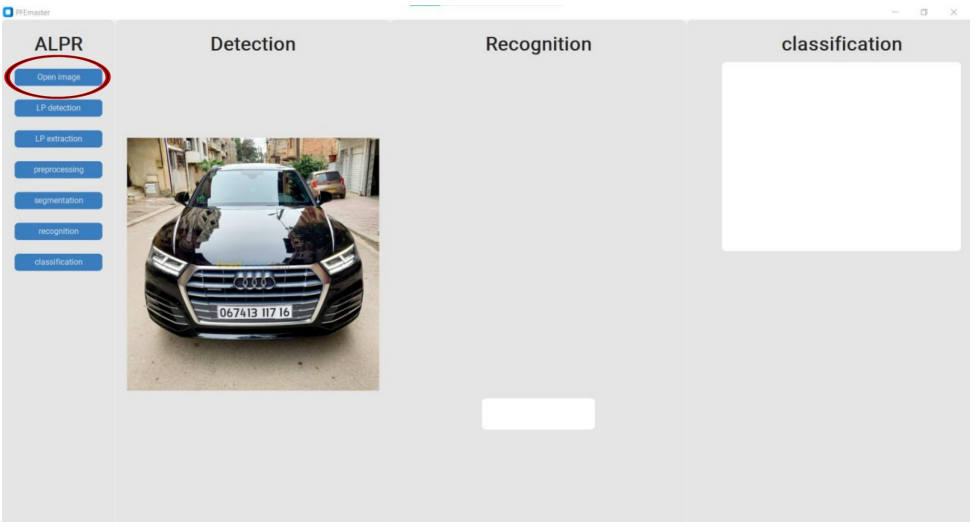

Figure 4.2: Open an image.

2. Once the image is successfully opened. By selecting the "LP detection" button, we initiate the process of detecting the license plate within the opened image. The YOLOv5 model, which has been previously trained on a large dataset, enables accurate and efficient license plate detection by leveraging advanced object detection techniques.The result shown in Figure 4.3.

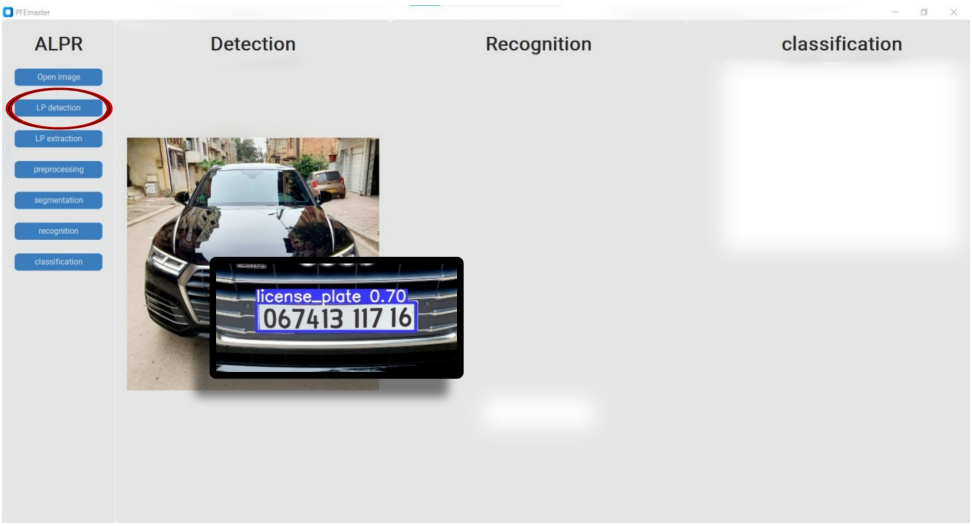

Figure 4.3: License plate detetion.

3. The "lp extraction" button initiates the process of extracting the license plate from the image with detection . It checks if the detected object is indeed a license plate and then proceeds to extract the corresponding portion of the image based on the bounding box coordinates using opencv library. The result shown in Figure 4.4.

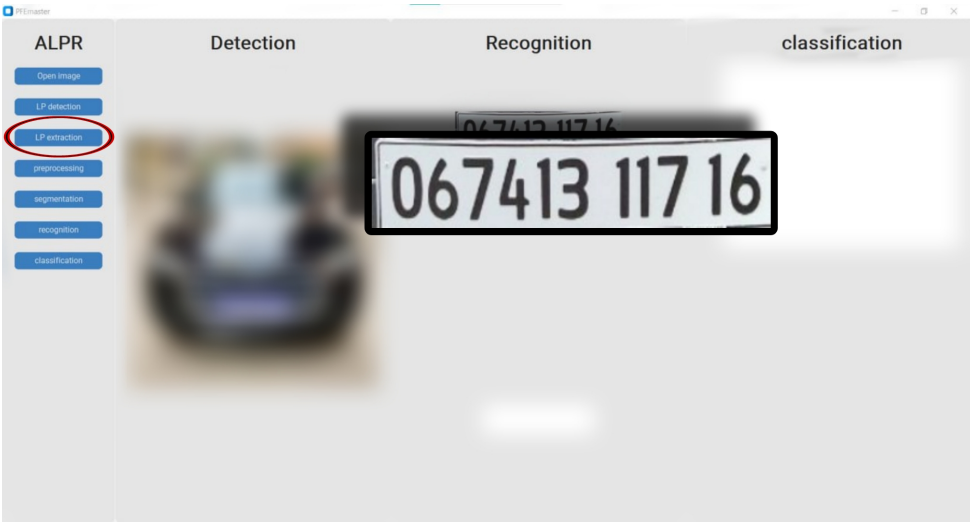

Figure 4.4: License plate extration.

4. The button "pre_processing" performs several image processing operations on the license plate image. It resizes the image, converts it to grayscale, applies Gaussian blur for noise reduction, performs adaptive thresholding to obtain a binary image, applies morphological operations for character enhancement, and straightens the license plate if needed using

Hough line transform and rotation.

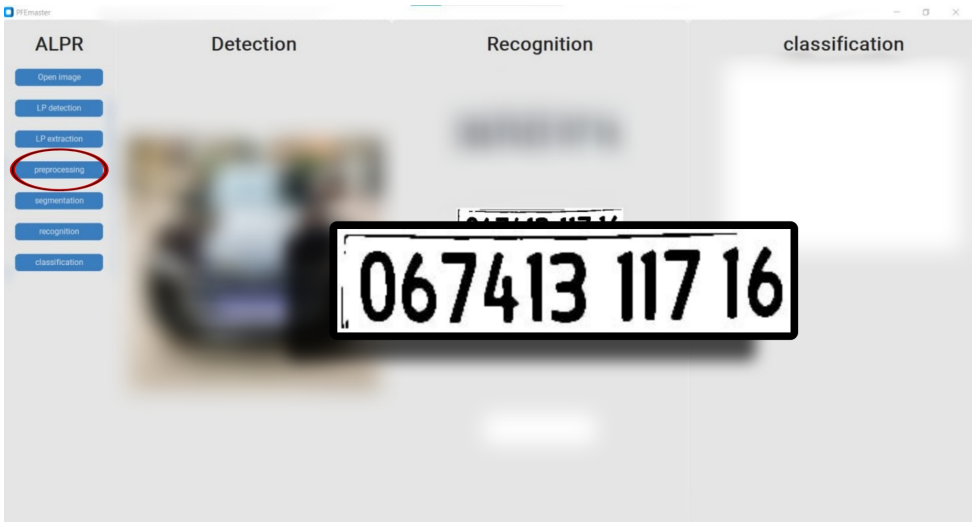

Figure 4.5: License plate pre_processed.

5. The "segmentation" button initiates the process of extracting individual characters from a pre-processed license plate image. This is achieved by identifying contours within the image that meet specific size criteria. The result shown in Figure 4.6.

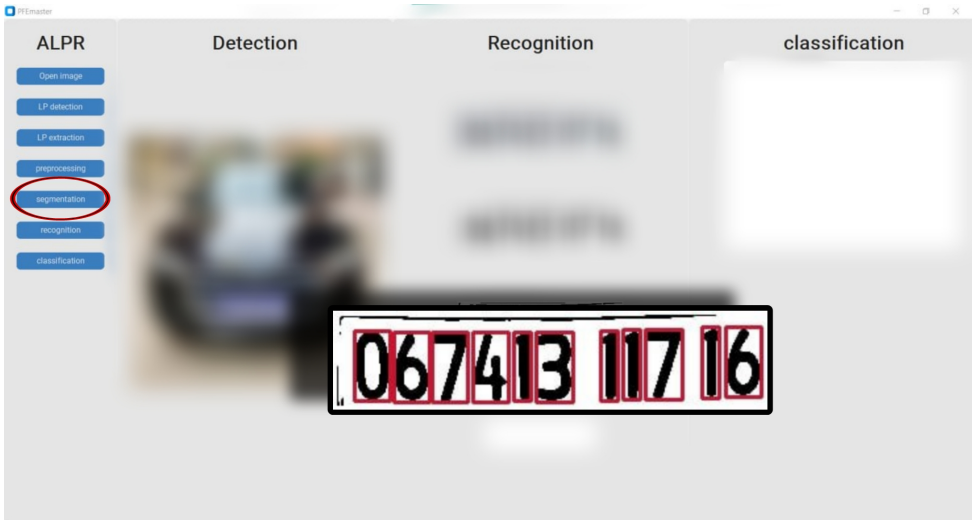

Figure 4.6: Character segmentation.

6. The "Recognition" button initiates the prediction process for the segmented characters using a trained Convolutional Neural Network (CNN) model. It iterates over the predicted images, preparing them by resizing and reshaping them to match the model's input requirements. The model predicts the class probabilities for each character image, and the character with the highest probability is chosen. By combining the predicted characters, it constructs the resulting license plate number. This plate number is saved to a text file and simultaneously displayed in a text box as shown in 4.7.

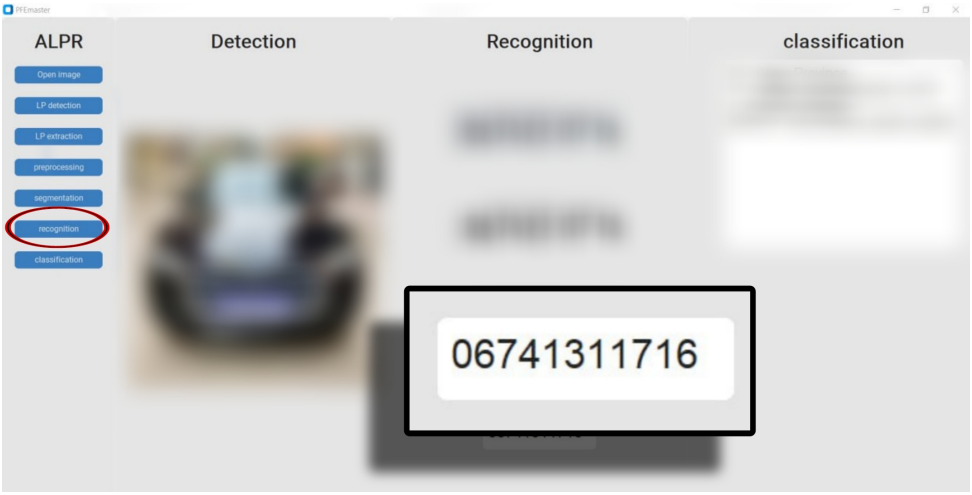

Figure 4.7: Character Recognition.

7. The "Classification" button performs a license plate classification process. It reads license plate numbers from a file and extracts information from different databases based on the plate numbers. It retrieves translations for the last two digits, the year of the license plate, and the type of vehicle. The extracted information is then displayed in a text box, providing details about the license plate, including its translation, year, type, and remaining characters. as shown in Figure 4.8.And also we have an example of non-Algerian plate in Figure 4.9.

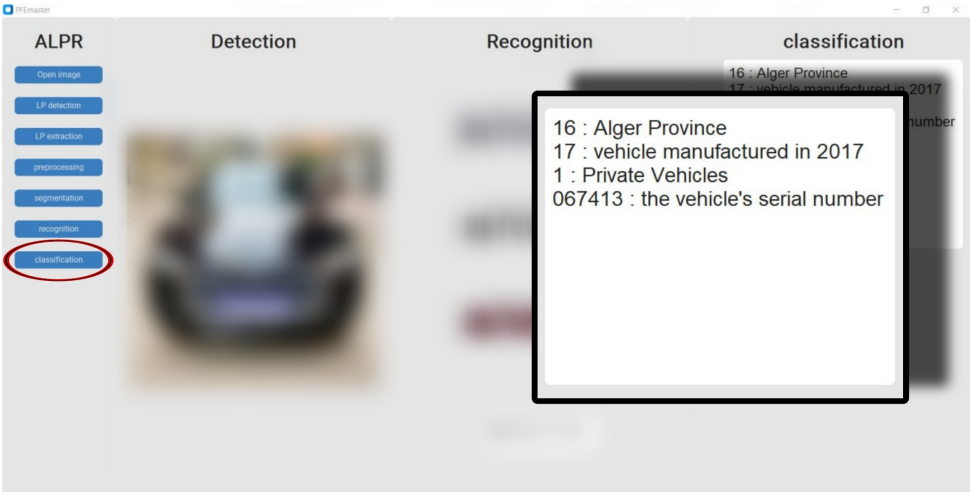

Figure 4.8: Character Classification.

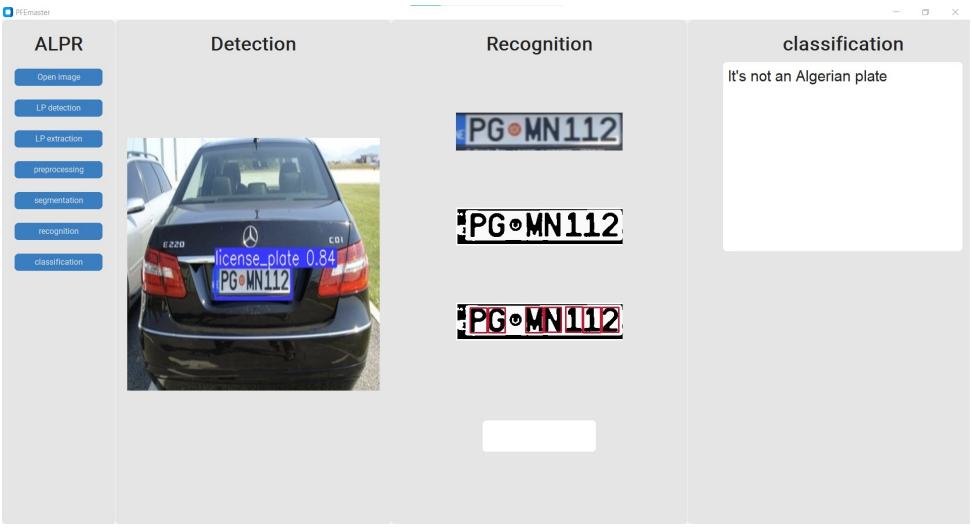

Figure 4.9: Example of non-Algerian plate.

4.7 Results analysis

Here we present both results of license plate detection and recognition tasks:

4.7.1 Results of detection model

Evaluation indicators When evaluating the performance of the YOLOv5 object detection algorithm, several indicators are commonly used. These indicators help measure the accuracy and efficiency of the model. Here are some key evaluation indicators for YOLOv5:

- Precision: Precision measures the proportion of correctly predicted bounding boxes for objects compared to the total predicted bounding boxes. It indicates the algorithm's ability to minimize false positives.

- Recall: Recall calculates the proportion of correctly predicted bounding boxes for objects compared to the total number of ground truth bounding boxes. It indicates the algorithm's ability to detect objects and minimize false negatives.

- Average Precision (AP): AP is a widely used metric that combines precision and recall. It measures the overall accuracy of the model across different object categories by considering precision-recall curves. Mean Average Precision (mAP) is the average AP value across multiple object categories.

In our system we have used YOLO v5s. a miniature version of YOLOv5, for detecting number plates in images. we explaned the steps in the previous chapter . The training result of the model is shown in Table 4.2 and Figure 4.10 .

Precision	**Recall**	**mAP50**
0.87	0.84	0.87

Table 4.2: The training result statistics.

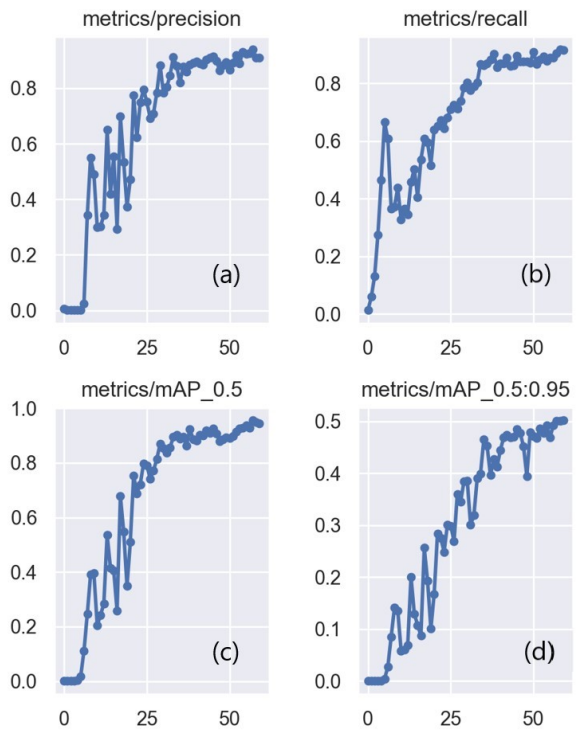

Figure 4.10: The training results of YOLOv5.

From (a) and (c) in Figure 4.3 , 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.

Figure 4.11: The output of running model on a test image.

4.7.2 Results of recognition model

The model achieved a validation loss of 0.3548, indicating that it effectively minimized the discrepancy between the predicted and actual values. This suggests that the model's predictions are generally close to the ground truth. Furthermore, the validation accuracy of 0.9386 reveals that the model correctly classified a significant proportion of the validation samples. These results indicate that the model exhibits a high level of accuracy in distinguishing between different classes or categories. Overall, the evaluation results emphasize the efficacy and potential of the model in solving the problem at hand, validating its capability to make accurate predictions and contributing to the overall success of your research. The accuracy and loss curve is shown in Figure 4.12 and Figure 4.13.

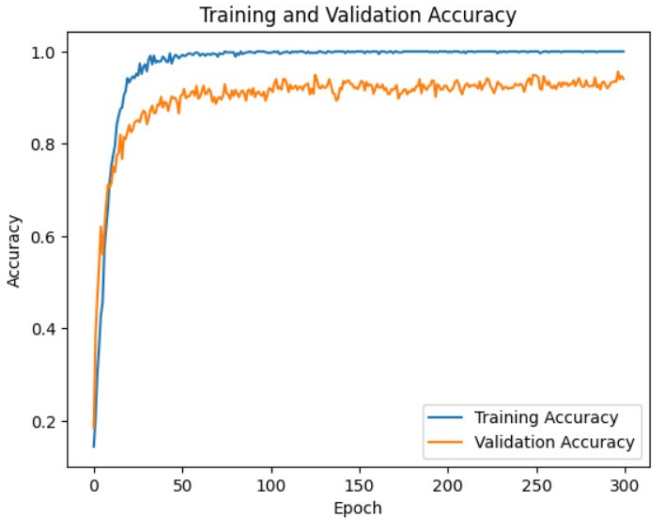

Figure 4.12: Accuracy Curve.

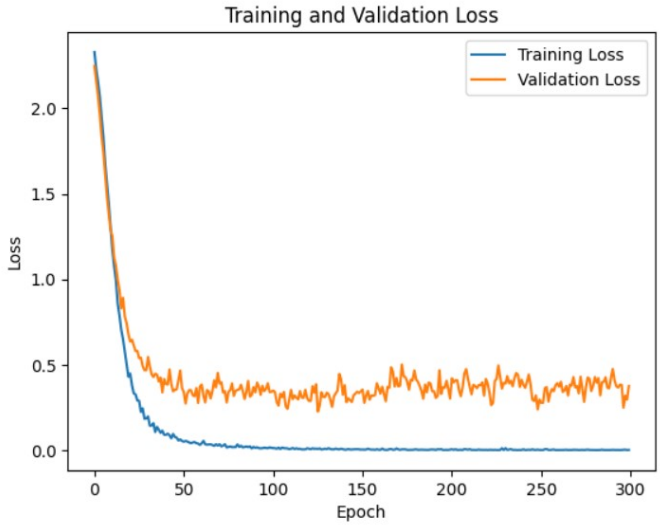

Figure 4.13: Training Loss Curve.

The graph below presents character recognition results from 50 license plates. The x-axis represents the characters found in the license plates, while the y-axis shows the count or percent-

age of occurrences. The graph indicates two categories: correct recognition and incorrect recognition. The correct recognition category reveals accurately identified characters, demonstrating the system's high accuracy in classifying and recognizing license plate characters. Conversely, the incorrect recognition category represents characters that were not accurately recognized by the system, although occurrences are minimal according to the graph.

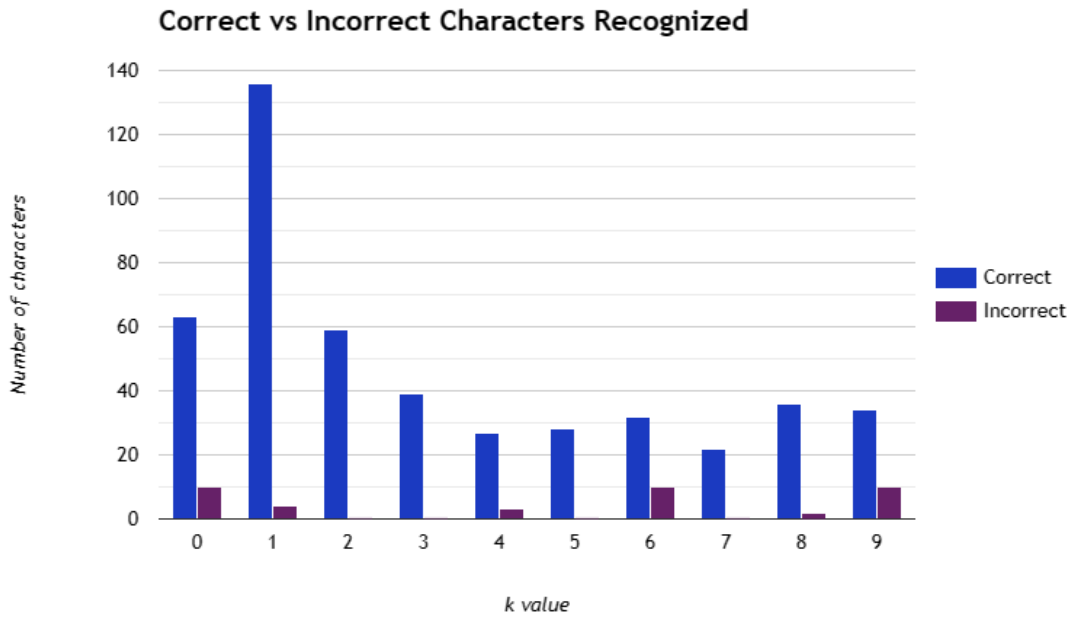

Figure 4.14: Correct vs incorrect characters recognized

From the graph, it is evident that the number of correctly recognized characters significantly outweighs the number of incorrectly recognized characters. The graph illustrates that the license plate recognition system achieves a high level of accuracy, as the instances of incorrect character recognition are minimal or almost non-existent. This observation highlights the effectiveness and robustness of the system in accurately identifying and classifying license plate characters. The graph's visual representation further reinforces the system's overall performance and provides confidence in its ability to consistently recognize license plate characters with a high degree of accuracy.

4.8 Discussion

From the results, it is evident that our system performs well in both the stages of detection and recognition, particularly in normal cases. The high precision and recall values indicate that the system successfully identifies and classifies license plates with a high level of accuracy. This implies that the system is effective in detecting and recognizing license plates 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 license plate 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 of license plates with high accuracy and efficiency.
- Our system excels in both license plate detection and recognition, even in challenging conditions such as skew problems and varying illuminations.
- Our system demonstrates high precision in license plate detection, ensuring accurate localization of 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 requires a substantial amount of time to complete.
- If we initiate the training for a second time, it does not complete the learning on the final checkpoint.Despite multiple attempts, the training process takes a significant amount of time but fails to reach completion.
- In the segmentation phase, we encounter challenges where certain characters may not be correctly segmented due to character connections or the presence of stickers or other non-relevant characters on the license plate.

4.9 Conclusion

In this chapter, we present the implementation details of our license plate detection and recognition system. For the license plate detection task, we utilize the YOLOv5 model, which has shown remarkable performance in object detection. Additionally, we employ a CNN model for the license plate recognition task. Through extensive experimentation and fine-tuning, we achieve impressive precision and accuracy results, demonstrating the effectiveness of our system.

General Conclusion

Automatic license plate detection and recognition systems are increasingly sought after due to their ability to enhance security, improve traffic management, streamline parking systems, enable efficient toll collection, and support smart city initiatives. In the context of Algerian license plates, our developed system specifically caters to the unique characteristics and requirements of this region.

We have meticulously designed a robust and efficient system for automatic Algerian license plate detection and recognition. Leveraging advancements in deep learning techniques, our system integrates the YOLOv5 algorithm, which excels in license plate detection, with a customized Convolutional Neural Network (CNN) model tailored for accurate plate recognition. This powerful combination allows our system to achieve remarkable precision in identifying and processing Algerian license plates.

Through extensive experimentation and validation, our system has demonstrated superior performance in Algerian license plate detection and recognition tasks. It exhibits high levels of accuracy, efficiency, and robustness, making it suitable for various real-world applications such as traffic surveillance, vehicle identification, parking management, and law enforcement in Algeria.

In our future works, we have outlined several perspectives and goals to further enhance our system for Algerian license plate detection and recognition. These include:
Training the model with a large dataset: While our current system has been trained on a comprehensive dataset of Algerian license plates, we recognize the importance of continuously expanding the dataset.

Enhancing segmentation phases: The segmentation phase is a critical step in license plate recognition. In our future works, we aim to refine and optimize the segmentation algorithms used in our system.

Achieving real-time processing: In our future works, we intend to optimize our system's algorithms and hardware infrastructure to achieve real-time performance.

By focusing on these perspectives in our future works, we aim to elevate the performance and capabilities of our system for Algerian license plate detection and recognition.

Bibliography

[1] Ouedkniss: Algeria's online marketplace. `https://www.ouedkniss.com/`. Accessed: April , 2023.

[2] Arohan Ajit, Koustav Acharya, and Abhishek Samanta. A review of convolutional neural networks. In *2020 international conference on emerging trends in information technology and engineering (ic-ETITE)*, pages 1–5. IEEE, 2020.

[3] Nur-A Alam, Mominul Ahsan, Md Abdul Based, and Julfikar Haider. Intelligent system for vehicles number plate detection and recognition using convolutional neural networks. *Technologies*, 9(1):9, 2021.

[4] Andrewmvd. Car Plate Detection Dataset. `https://www.kaggle.com/datasets/andrewmvd/car-plate-detection`. Accessed: March, 2023.

[5] Naga Surya Sandeep Angara. Automatic license plate recognition using deep learning techniques. 2015.

[6] Samiul Azam and Md Monirul Islam. Automatic license plate detection in hazardous condition. *Journal of Visual Communication and Image Representation*, 36:172–186, 2016.

[7] Nikoletta Bassiou and Constantine Kotropoulos. Color image histogram equalization by absolute discounting back-off. *Computer Vision and Image Understanding*, 107(1-2):108–122, 2007.

[8] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[9] Eli Bressert. Scipy and numpy: an overview for developers. 2012.

[10] Rongbao Chen and Yunfei Luo. An improved license plate location method based on edge detection. *Physics Procedia*, 24:1350–1356, 2012.

[11] Masahiro Daibo. Toroidal vector-potential transformer. In *2017 Eleventh International Conference on Sensing Technology (ICST)*, pages 1–4. IEEE, 2017.

[12] Shubhada Deshmukh, Manasi Patwardhan, and Anjali Mahajan. Survey on real-time facial expression recognition techniques. *Iet Biometrics*, 5(3):155–163, 2016.

[13] Shan Du, Mahmoud Ibrahim, Mohamed Shehata, and Wael Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on circuits and systems for video technology*, 23(2):311–325, 2012.

[14] N Eswar and D Gowri Shankar Reddy. Morphological operation based vehicle number plate detection. *International Journal of Engineering Research And*, 9(2):428–433, 2020.

[15] Faizan387. Car Number Plate Recognition Repository. https://github.com/faizan387/Car-Number-Plate-Recognition/tree/master/Car%20plate%20recognition%20Punjab/data. Accessed: April , 2023.

[16] M Icaza. Tensorflowsharp: Tensorflow api for .net languages. *Accessed: January*, 18:2019, 2019.

[17] Jupyter Project. Jupyter Documentation. https://jupyter.org/. Accessed: june, 2023.

[18] Keras Team. Keras Documentation. https://keras.io/. Accessed: june, 2023.

[19] Julia Krasevec, Xiaoyi An, Richard Kumapley, France Bégin, and Edward A Frongillo. Diet quality and risk of stunting among infants and young children in low-and middle-income countries. *Maternal & child nutrition*, 13:e12430, 2017.

[20] Rayson Laroca, Evair Severo, Luiz A Zanlorensi, Luiz S Oliveira, Gabriel Resende Gonçalves, William Robson Schwartz, and David Menotti. A robust real-time automatic license plate recognition based on the yolo detector. In *2018 international joint conference on neural networks (ijcnn)*, pages 1–10. IEEE, 2018.

[21] Rayson Laroca, Luiz A Zanlorensi, Gabriel R Gonçalves, Eduardo Todt, William Robson Schwartz, and David Menotti. An efficient and layout-independent automatic license plate recognition system based on the yolo detector. *IET Intelligent Transport Systems*, 15(4):483–503, 2021.

[22] Hui Li, Peng Wang, and Chunhua Shen. Toward end-to-end car license plate detection and recognition with deep neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):1126–1136, 2018.

[23] Hui Li, Peng Wang, Mingyu You, and Chunhua Shen. Reading car license plates using deep neural networks. *Image and Vision Computing*, 72:14–23, 2018.

[24] Yujie Liu, He Huang, Jinde Cao, and Tingwen Huang. Convolutional neural networks-based intelligent recognition of chinese license plates. *Soft Computing*, 22(7):2403–2419, 2018.

[25] Weidong Min, Xiangpeng Li, Qi Wang, Qingpeng Zeng, and Yanqiu Liao. New approach to vehicle license plate location based on new model yolo-l and plate pre-identification. *IET Image Processing*, 13(7):1041–1049, 2019.

[26] Rahul Mishra, Hari Prabhat Gupta, and Tanima Dutta. A survey on deep neural network compression: Challenges, overview, and solutions. *arXiv preprint arXiv:2010.03954*, 2020.

[27] Sérgio Montazzolli and Claudio Jung. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In *2017 30th SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, pages 55–62. IEEE, 2017.

[28] Jojo Moolayil, Jojo Moolayil, and Suresh John. *Learn Keras for deep neural networks*. Springer, 2019.

[29] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[30] IKRAME NOUAR. La détection des attaques botnet dans l'industrie internet des objets (iiot). 2022.

[31] T Nukano, M Fukumi, and M Khalid. Vehicle license plate character recognition by neural networks. In *Proceedings of 2004 International Symposium on Intelligent Signal Processing and Communication Systems, 2004. ISPACS 2004.*, pages 771–775. IEEE, 2004.

[32] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[33] PIL Contributors. Python Imaging Library (PIL) Documentation. `https://pillow.readthedocs.io/en/stable/`. Accessed: june, 2023.

[34] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36, 2018.

[35] Irina Valeryevna Pustokhina, Denis Alexandrovich Pustokhin, Joel JPC Rodrigues, Deepak Gupta, Ashish Khanna, K Shankar, Changho Seo, and Gyanendra Prasad Joshi. Automatic vehicle license plate recognition using optimal k-means with convolutional neural network for intelligent transportation systems. *Ieee Access*, 8:92907–92917, 2020.

[36] Python Software Foundation. Python. `https://www.python.org/`. Accessed on 10th June 2023.

[37] AM Raid, WM Khedr, MA El-Dosuky, and Mona Aoud. Image restoration based on morphological operations. *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, 4(3):9–21, 2014.

[38] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[39] Saif Ur Rehman, Moiz Ahmad, Asif Nawaz, and Tariq Ali. An efficient approach for vehicle number plate recognition in pakistan. *The Open artificial intelligence journal*, 6(1), 2020.

[40] Harshavardhan Seetha, Vimal Tiwari, Kartik Reddy Anugu, DS Makka, and DR Karnati. A gui based application for pdf processing tools using python & customtkinter. *Int. J. Res. Appl. Sci. Eng. Technol.*, 2023.

[41] 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)*, volume 1, pages 1132–1138. IEEE, 2017.

[42] MM Shaifur Rahman, Mst Shamima Nasrin, Moin Mostakim, and Md Zahangir Alom. Bangla license plate recognition using convolutional neural networks (cnn). *arXiv e-prints*, pages arXiv–1809, 2018.

[43] Jithmi Shashirangana, Heshan Padmasiri, Dulani Meedeniya, and Charith Perera. Automated license plate recognition: a survey on methods and techniques. *IEEE Access*, 9:11203–11225, 2020.

[44] Hengliang Shi and Dongnan Zhao. License plate recognition system based on improved yolov5 and gru. *IEEE Access*, 11:10429–10439, 2023.

[45] 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)*, pages 580–596, 2018.

[46] Ibtissam Slimani, Abdelmoghit Zaarane, Abdellatif Hamdoun, and Issam Atouf. Vehicle license plate localization and recognition system for intelligent transportation applications. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1592–1597. IEEE, 2019.

[47] Farhana Sultana, Abu Sufian, and Paramartha Dutta. A review of object detection models based on convolutional neural network. *Intelligent computing: image processing based applications*, pages 1–16, 2020.

[48] TensorFlow. TensorFlow-Keras Documentation. `https://www.tensorflow.org/guide/keras?hl=fr`. Accessed: April, 2023.

[49] Sandro Tosi. *Matplotlib for Python developers*. Packt Publishing Ltd, 2009.

[50] Ultralytics. YOLOv5 Repository. `https://github.com/ultralytics/yolov5`. Accessed: April , 2023.

[51] Abdul Vahab, Maruti S Naik, Prasanna G Raikar, and SR Prasad. Applications of object detection system. *International Research Journal of Engineering and Technology (IRJET)*, 6(4):4186–4192, 2019.

[52] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[53] Frederick M Waltz and John WV Miller. Efficient algorithm for gaussian blur using finite-state machines. In *Machine Vision Systems for Inspection and Metrology VII*, volume 3521, pages 334–341. SPIE, 1998.

[54] Yuh-Rau Wang, Wei-Hung Lin, and Shi-Jinn Horng. A sliding window technique for efficient license plate localization based on discrete wavelet transform. *Expert Systems with Applications*, 38(4):3142–3146, 2011.

[55] D Wazalwar, E Oruklu, and J Saniie. Design flow for robust license plate localization. In *2011 IEEE International Conference on Electro/Information Technology*, pages 1–5. IEEE, 2011.

[56] B-F Wu, S-P Lin, and C-C Chiu. Extracting characters from real vehicle licence plates out-of-doors. *IET Computer Vision*, 1(1):2–10, 2007.

[57] Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 396:39–64, 2020.

[58] Lele Xie, Tasweer Ahmad, Lianwen Jin, Yuliang Liu, and Sheng Zhang. A new cnn-based method for multi-directional car license plate detection. *IEEE Transactions on Intelligent Transportation Systems*, 19(2):507–517, 2018.

[59] Hong-ke Xu, Fu-hua Yu, Jia-hua Jiao, and Huan-sheng Song. A new approach of the vehicle license plate location. In *Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)*, pages 1055–1057. IEEE, 2005.

[60] Qingchen Zhang, Laurence T Yang, Zhikui Chen, and Peng Li. A survey on deep learning for big data. *Information Fusion*, 42:146–157, 2018.

[61] Xinyi Zhou, Wei Gong, WenLong Fu, and Fengtong Du. Application of deep learning in object detection. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 631–634. IEEE, 2017.

[62] Li Zou, Meng Zhao, Zhengzhong Gao, Maoyong Cao, Huarong Jia, and Mingtao Pei. License plate detection with shallow and deep cnns in complex environments. *Complexity*, 2018:1–6, 2018.

[63] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

[64] Rodolfo Zunino and Stefano Rovetta. Vector quantization for license-plate location and image coding. *IEEE Transactions on Industrial Electronics*, 47(1):159–167, 2000.