A Major Project Report on

AUTOMATIC AND FAST VEHICLE NUMBER PLATE DETECTION USING NEURAL NETWORKS

Submitted In partial fulfillment of the Requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

BY

PENTAM RAJASEKHAR	**(17UJ1A0516)**
SUGUREDDY SINDHU REDDY	**(17UJ1A0542)**
GANDAM SUCHARITHA	**(17UJ1A0544)**
BANDA APOORVA REDDY	**(17UJ1A0526)**
MOHD ATIF UDDIN	**(17UJ1A0514)**

Under the Esteemed Guidance of

Mrs. K. ASHWINI M.Tech.

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MALLA REDDY COLLEGE OF ENGINEERING AND MANAGEMENT SCIENCE
(Approved by AICTE New Delhi & Affiliated to JNTU Hyderabad)
Kistapur, Medchal, Medchal Dist- 501401.

2017-2021

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
HYDERABAD, KUKATPALLY, HYDERABAD-85.

MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT SCIENCES

(Approved by AICTE New Delhi & Affiliated to JNTU Hyderabad)
Kistapur, Medchal, Medchal Dist- 501401.

CERTIFICATE

This is to certify that the major project report entitled "Automatic and Fast Vehicle Number Plate Detection Using Neural Networks" being submitted by

PENTAM RAJASEKHAR	**(17UJ1A0516)**
SUGUREDDY SINDHU REDDY	**(17UJ1A0542)**
GANDAM SUCHARITHA	**(17UJ1A0544)**
BANDA APOORVA REDDY	**(17UJ1A0526)**
MOHD ATIF UDDIN	**(17UJ1A0514)**

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Jawaharlal Nehru Technological University , Hyderabad, during the academic year 2017-2021 is a record of bonafied work carried out under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Internal Guide

Mrs.K.Ashwini M.Tech

Assistant Professor
Department.

Head of The Department

Dr. D .MAHAMMAD RAFI, M.Tech, Ph.D

Professor & Head of CSE
Department of CSE

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost, I express my deep debt of gratitude to the Chairperson and Managing Trustee **Mr.Malla Reddy** for their immense contribution in making this organization grow and providing me the state of the art facilities to do this project work.

My heartfelt gratefulness to Director of MREM **Mr. L.Venugopal Reddy** for their deep commitment and dedication, to bring this institution to the peak in terms of discipline and values.

I owe my full satisfaction in my project work to the extensive hands of cooperation of Principal Academics **Dr.CNV.SRIDHAR** , M.Tech, Ph.D.

I would like to express my deep appreciation and sincere gratitude to my project internal guide, **Mrs.K.Ashwini** M.Tech, Assistant Professor, Department of Computer Science & Engineering, Malla Reddy Engineering College and Management Sciences, for her guidance, support, encouragement, understanding and patience. I have been honored to work under her supervision and learn from her advice and useful insights throughout my project work.

I extend my gratitude to **Dr. D.Mahammad Rafi**, Professor and Head, Department of Computer Science and Engineering for his constant support to complete the project work.

I express my sincere thanks to all the teaching and non-teaching staffs of Malla Reddy Engineering College and Management Sciences who cooperated with me, which helped me to realize my dream.We would like to thank our internal project mates and department faculties for their full-fledged guidance and giving courage to carry out the project.I am very much thankful to one and all that helped me for the successful completion of my project.

PENTAM RAJASEKHAR (17UJ1A0516)
SUGUREDDY SINDHU REDDY (17UJ1A0542)
GANDAM SUCHARITHA (17UJ1A0544)
BANDA APOORVA REDDY (17UJ1A0526)
MOHD ATIF UDDIN (17UJ1A0514)

ABSTRACT

The main aim of License Plate Recognition system is that, it a real-time embedded system which automatically recognizes the license plate of vehicles. There are many applications ranging from complex security systems to common areas and from parking admission to urban traffic control. Automatic license plate recognition (ALPR) has complex characteristics due to diverse effects such as light and speed. Most of the license plate recognition systems are built using proprietary tools like Matlab which takes a long process and time and also does have several limitations. This idea presents an alternative method of implementing ALPR systems using Free Software including Python and the Open Computer Vision Library.

This project deals with problematic from field of artificial intelligence, machine vision and neural networks in construction of an automatic number plate recognition system (ANPR). This project includes brief introduction of automatic number plate recognition, which ensure a process of number plate detection, processes of proper characters segmentation, normalization and recognition. In this project, the task of recognizing number plate is considered. First the image of number plate is captured by camera. Number plate is segmented by using horizontal and vertical projection. After that feature extraction techniques are used to extract the characters from segmented data. Neural Network algorithms are used to recognize the characters which improve the color and brightness.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ALPR	**Automatic License Plate Recognition**
OCR	**Optical Character Recognition**
CNN	**Convolutional Neural Network**
ANPR	**Automatic Number Plate Recognition**
ReLU	**Rectified Linear Units**
RFID	**Radio-Frequency Identification**
OpenCV	**Open Source Computer Vision Library**
HR	**High Resolution**
LP	**License Plate**
CCL	**Connected Component Labelling**
MLP	**Multi-Layer Perceptron**
DAST	**Dynamic Application Security Testing**
GPU	**Graphics Processing Units**

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

With rising traffic on roads and number of vehicles on roads, it is getting very difficult to manually handle laws, traffic rules and regulations for smooth traffic moment. Toll-booths are installed on freeways and parking complex, where the vehicle has to stop to pay the toll or parking charge fees. Also, Traffic maintenance systems are constructed on freeways to analyze for vehicles moving at speeds not allowed by law. All these operations have a scope of improve development. In the center of all these processes lies a vehicle. The vital question here is how to find a particular vehicle? The obvious response to this question is by using the vehicle's number plate. This number differentiate one vehicle from the other, which is effective especially when both are of same type of make and model.

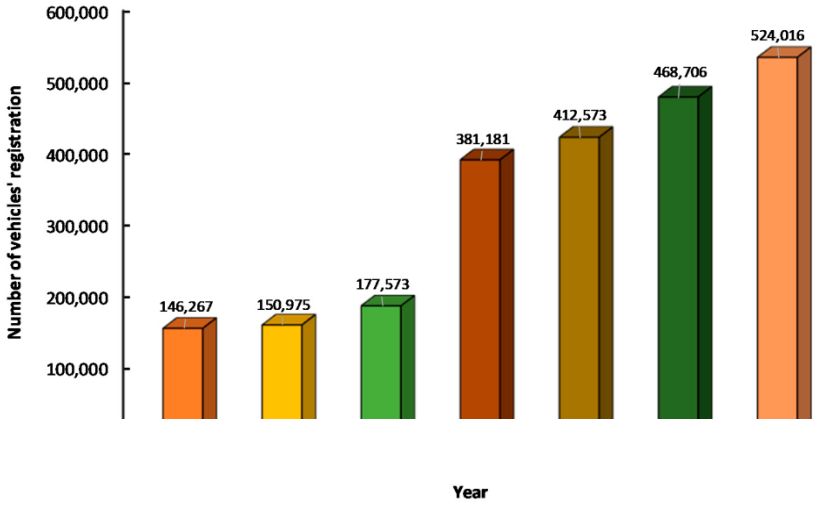


Figure 1.1- Registration of vehicles' number in India in the past eight years

An automated system can be developed to find the license plate of a vehicle and recognize the characters from the region having a license plate. The vehicle license plate number which can be utilized to fetch farther information data about the vehicle and its owner, which can be used for further processing.

The live CCTV footage is converted into frames. The frames are passed through OPENCV algorithm to detect the cars in it. The detected cars are stored in separate

images in a folder. These images are checked for number plate. The detected number plate will be cropped and stored in another folder. The characters in these number plates are recognized using OCR (Optical Character Recognition). The extracted text is then copied to an excel sheet with the time, date and vehicle number. This system tends to give a higher accuracy than the existing systems and has an additional advantage of being implemented in real time. Machine Learning which uses multiple layers to get high level features from a raw input.

Deep Learning is now used in almost all the real time applications. Unlike other algorithms, it shows a high level of accuracy and minimum acceptable errors. This system uses Convolutional Neural Network (CNN) to detect the cars and number plate. The main aim of the system is to design a deep learning model that could read the number plate of the fast-moving vehicles on the road using the surveillance camera and save the extracted number plate data.

1.2 AIM AND OBJECTIVE

The aim of this project is to be able to develop a system for automatically recognizing Indian number plates from High Resolution (HR) digital images by combining advanced Computer Vision techniques with some of the most powerful Machine Learning algorithms.

Hence, the most important objectives within the project are:

- To analyse the different existing Computer Vision techniques used in Automatic number plate recognition(ANPR) in order to select the most efficient and appropriate ones. To learn to use these techniques and make the most of their capabilities.
- To analyse the different Machine Learning methods and choose the most suitable ones for ANPR. To learn how they work and to adapt them to the ANPR application.
- To develop an efficient, fast and reliable ANPR application that is able to compete with other recently developed ANPR systems of similar characteristics.
- Apart from the ANPR application itself, to develop another separate application that quantitatively analyses the performance of the ANPR system.

The major objective of this work is to design an Indian vehicle number plate

detection and recognition using OpenCV and Python to deal with the following challenges:

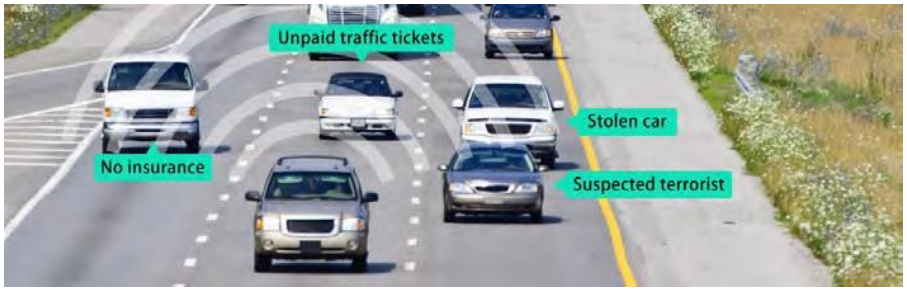
Dealing with varying illuminated images.

- Dealing with bright and dark objects.
- Dealing with noisy images.
- Dealing with non-standard number plates.
- Dealing with cross-angled or skewed number plates.
- Dealing with partially worn our number plates.

1.3 BACKGROUND AND MOTIVATION

Automatic Number Plate Recognition is a computer vision technology that efficiently identifies vehicle number plates from images without the need for human intervention. In recent years, it has become more and more important due to three main factors: the growing number of cars on the roads, the rapid development of image processing techniques and the great quantity of real-life applications that this technology offers.

Some of the most typical applications of ANPR systems are traffic law enforcement, automatic toll collection or parking lot access control. But this technology is also widely used for other, perhaps, more inspiring purposes like crimes resolution, as it helps to identify the cars of the offenders. In Figure 1.2, some other examples of these applications can be appreciated.



1.2 -Examples of ANPR Applications

However, the development of ANPR systems is no easy task, since it faces numerous challenges due to environmental and number plate variations. As for the former, varying illumination or background patterns greatly affect number plate recognition. In effect, varying illumination can degrade the quality of the car image and background patterns add extra difficulty to the number plate location process. And, as for the latter, the location, quantity, size, font, colour or inclination of number plates constitute very challenging factors in the development of a consistent ANPR system, which is clearly illustrated in the following section.

1.4 NUMBER PLATES IN INDIAN UNION

1.4.1 Format of number plates

- Private vehicles:
  - ❖ Private vehicles, by default, have black lettering on a white background (e.g. **AP 02 BK 1084**).
  - ❖ Vehicles which run purely on electricity have white lettering on a green background (e.g. **AP21BP7331**)
- Commercial vehicles:
  - ❖ Commercial vehicles such as taxis, buses and trucks, by default, have black lettering on a yellow background (e.g. **UP 19 D 0343**).
  - ❖ Vehicles available on rent for self-drive have yellow lettering on a black background (e.g. **KA 08 J 9192**).
  - ❖ Vehicles which run purely on electricity have yellow lettering on a green background (e.g. **MH 12 RN 1289**)
- Vehicles belonging to foreign missions:
  - ❖ Vehicles registered to an embassy or United Nations have white lettering on a light blue background (e.g. **199 CD 1** and **23 UN 1**[2] respectively).
  - ❖ Vehicles registered to a consulate have black lettering on a yellow background (e.g. **199 CC 999**).

- Vehicles registered by Indian Armed Forces have white lettering starting with an Arrow on a black background (e.g. ↑03D 153874H).

1.4.2 Temporary Registration

- Unsold vehicles belonging to a vehicle manufacturer or a dealer have white lettering on a red background (e.g. HR 26 TC 7174).
- Sold vehicles awaiting a permanent registration have red lettering on a yellow background (e.g. TS 07 D TR 2020).

1.4.3 Current Format

There are 4 parts of the registration index's current format and they are as follows:

- The 1st 2 letters indicate Union Territory or the state to which the motor vehicle has been registered.
- The next 2 digit numbers are sequential number of the district. Because of heavy volume of the registration, the numbers are given to RTO offices as well of registration.
- The 3rd part has a 4 digit number on each plate. A letter or letters is prefixed at the time when the 4 digit number goes out and then 2 letters.
- The 4th part is the international oval 'IND' having a Hologram at the top with a Chakra.

There are some advantages of this numbering scheme which are:

- The district of state of registration of a vehicle.
- In case of any police investigation of a vehicle-related crime or an accident, witnesses remember the initial code letters of the area usually. It then becomes simple to shortlist the suspect vehicles to a smaller number and check the database without knowing the full number.

1.4.4 Special Formats

In some states like Bihar, Gujarat and Delhi, the initial 0 digit is omitted of the district code, and so Delhi district 2 digits looks like DL 2 not DL 02, BR 8 not BR 08, GJ 5 not GJ 05.

The Delhi state is having an additional code in its registration code:

DL 11 CAA 1111

Where DL is the 2 letter code for Delhi (DL), the additional letter C is for the category of vehicle like S is for 2-wheelers, C is for SUVs and cars, P for public vehicles like buses, Y for hire vehicles, V for the pick-up vans and trucks, T is for tourist licensed taxis and vehicles and R for the 3-wheeled rickshaws. This system is applicable in other states as well. Like for example, Rajasthan, where the 2 letter code is RJ, G for the goods vehicles, S for scooters and P for the passenger vehicles.

CHAPTER-2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

Shilpi Chauhan and Vishal Srivastava(2014). The study is about the Matlab based Vehicle Number Plate Recognition. Vehicle Number plate detection plays a significant role in detecting vast number of license plates. It detects the number plates of the vehicles and recognize the characters at government sectors. Most of the license plate recognition systems are built using proprietary tools like Matlab which takes a long process and time consuming and requires complex computations.

Manisha shirvoikar,Jairam Parab (2013). This paper presents an approach to license plate localization and segmentation. Vehicle license plate identification and segmentation is an important part of the vehicle license plate recognition. The proposed idea of work is based on image processing tool and helps in functions like vehicle license plate recognition. The proposed algorithm is based on a combination of morphological operation for license plate localization, extraction of plate region and segmentation of characters. But the disadvantage of this project is low resolution images are not relatively identified.

Sachin bhosale (2013). This research paper describes the automated toll collection system for toll gate based on Radio-Frequency Identification(RFID)technology. Most of the toll collection systems commonly used in Myanmar is manual transaction. Nowadays, streams of traffic are increased and toll gate on highways are congested. It will cause the traffic jam and waste time. The objective of this journal paper is to transform manual transaction to automated toll collection with the help of RFID. It is proposed as a low cost optimized solution using RFID and GSM mobile technology.

Amninder Kaur,Sonika Jindal (2012).License plate recognition using support vector machine. In this paper proposed approach is present. It has considered that the Indian number plates, where rear follows the number plate standards. This system consists of few algorithms life “Feature based number plate localization” for locating the number plate, “Image Scissoring” algorithm for character segmentation and proposed algorithm for character recognition using Support Vector Machine (SVM).System can

recognize single or double line number plate. The disadvantage of this project is it does not perform very well if the input contains relatively more noise.

S.no	Title	Author	Year	Description	Disadvantage
1	Matlab Based Vehicle Number Plate Recognition	Ms. Shilpi Chauhan and Vishal Srivastava	2014	Detects the number plate of the vehicles and recognize the characters at government sectors.	MATLAB is Costly, Time Consuming and requires complex Computations
2	Effective method of license plate localization and Segmentation of vehicles	Manisha Shirvoikar, Jairam Parab	2013	LPLS system to detect tracking vehicles for the purpose of parking admission, traffic management and law enforcement.	Low resolution images are not relatively identified
3	Automated toll plaza system using RFID	Sachin Bhosale	2013	It is proposed as a low cost optimized solution using RFID and GSM mobile technology.	Highly costly and Failed to Read High Dimentional Data
4	License Plate Recognition Using Support Vector Machine	AmninderKaur, Sonika Jindal	2012	(ALPR) system is used. Algorithm used "Feature based number plate Localization" for locating the number plate, "Image Scissoring".	It does not perform very well if the Input contains relatively more noise

Table2.1-Literature survey

In the Comparison of various vehicle number plate detection, as discussed in the Literature Survey, the neural network model performance seems to be more effective and is of less disadvantages than other models. So, in this project work neural network is considered as a better existing system and due to this reason, it has been used as a reference during the comparison of performance to vehicle number plate detection with high accuracy values.

The neural network model to initiate growth forecasting, which has the nature of "black box", increased calculations, prone to over fitting, and the empirical nature of model development. The amount of data required for normal operation of neural networks is very high, which may or may not be possible.

2.2 DISADVANTAGES OF EXISTING SYSTEM

- **More Expenditure**: License plate recognition systems are implemented using proprietary technologies like Matlab which are costly and not efficient.
- **Privacy Concerns:** Using ANPR cameras raises privacy concerns for many people, who dislike the idea of their data being stored for months.
- **More Manpower**: ANPR cameras may struggle to work in adverse weather conditions, such as heavy rain or snow, where the number plate is obscured or distorted.
- **Time Consuming**: Most of the license plate recognition systems are built using proprietary tools like Matlab which takes a long process and time.

2.3 SUMMARY OF EXISTING SYSTEM

This Literature Survey gives a brief study on how to detect vehicle number plates and detailed study on various research issues on ANPR. The various ANPR and different approaches on vehicle number plate detection have been analyzed. Automatic and fast vehicle number plate detection system is the most researched topic of neural networks. Although there are many other research topics that have been investigated in the literature, it is believed that this selected review has covered the most important aspects of neural networks in solving Automatic and fast vehicle number plate detection problems. Neural networks have been demonstrated to be a competitive alternative to Automatic and fast vehicle number plate detection for many practical problems. However, while neural networks have shown much promise, many issues still remain

unsolved or incompletely solved. Recently, several authors have pointed out the lack of the rigorous comparisons between neural network and other Automatic and fast vehicle number plate detection in the current literature. The Proposed Convolutional Neural Network algorithm using Deep Learning Techniques is the efficient algorithm to address these issues in .Automatic and fast vehicle number plate detection.

CHAPTER-3

SYSTEM ANALYSIS

3.1 PROBLEM STATEMENT

To overcome the problems occurred in existing system we design and develop a real-time detection, tracking and license plate recognition system that will work efficiently under the conditions of slow moving objects and the objects that are merged into the background due to a temporary stop and becoming foreground again, adaptive to different traffic environment conditions, robustness against progressive or sudden illumination changes, Occlusions, identification time of the system should be as short as possible. The system should detect all the types of vehicles, recognize all the license plates of the country and should also be resistant to any kinds of disturbances, which may occur in images and mechanical plate damages which may appear in reality.

Every system should fulfil the following demands:

- The identification time of the system should be as short as possible.
- The system should detect all the types of vehicles.
- The system should recognize all the license plates of the country.
- The system should also be unaffected to any kinds of disturbances, which may occur in images and mechanical plate damages which may appear in reality.

Other than this, there are several other parameters on which the quality of the recognition depends. These limitations are:

- High quality imaging – the quality of the captured image is high, so that it will be easy to evaluate the attributes of the image.
- Minimal skewing and rotation – the camera is fit such that the captured image or video will not suffer much decent angle of skewing and rotation.
- Better Flash – as the License plate is retro-reflective in nature, so if high illumination will be there, then the LP region will be much more brightened and the attributes of the license plate will not be much clearer.

3.2 PROPOSED SYSTEM

In India, basically, there are two kinds of license-plates, black characters in white plate and black characters in a yellow plate. The former for private vehicles and latter for commercial, public service vehicles. Our proposed system is to show that free and open source technologies are matured enough for scientific computing domains. Also, we have implemented noise reduction techniques to increase more accuracy in detection of numbers.

The system works satisfactorily for wide variations in illumination conditions and different types of number plates commonly found in India. It is definitely a better alternative to the existing proprietary systems, even though there are known restrictions with high resolution to detect the plate using OpenCV and Neural Networks which is most easy to understand and make changes.

With the aim to connect all the smart vehicle number plate detection systems and to improve cost efficiency and reduce error. we are proposing a system which can be effectively reduce number of human resources and cost and detects vehicle number plates in a split of a second with no or negligible error. This system can be used for multiple purposed with a minimum setup cost for the required hardware. This is all about the proposed system of the project "Automatic Vehicle Number Plate Recognition System Using Machine Learning".

The proposed system for license plate recognition is presented in this section. Input to the system is a vehicle image that has been acquired through image acquisition device and output is the editable form of license number.

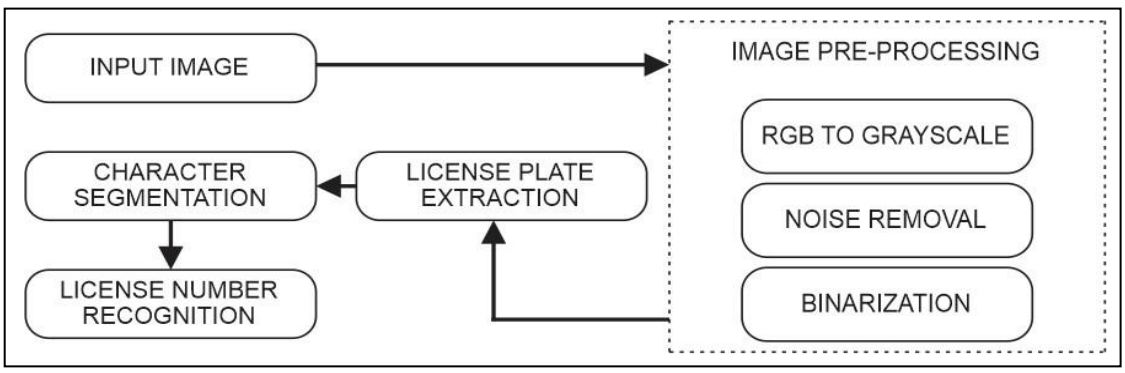


Figure 3.1-Proposed system of ANPR

3.2.1 Working Process

The flowchart of proposed system is shown in Figure 3.1 which consists of the following main steps:

1. Acquired Input Image
2. Image Pre-processing:

a. RGB to grayscale conversion

b. Noise removal by Iterative Bilateral Filtering

c. Image binarization

3. License plate extraction using Sobel's edge detection algorithm/Smearing algorithm
4. Character Segmentation
5. Recognition of License number using convolution neural network

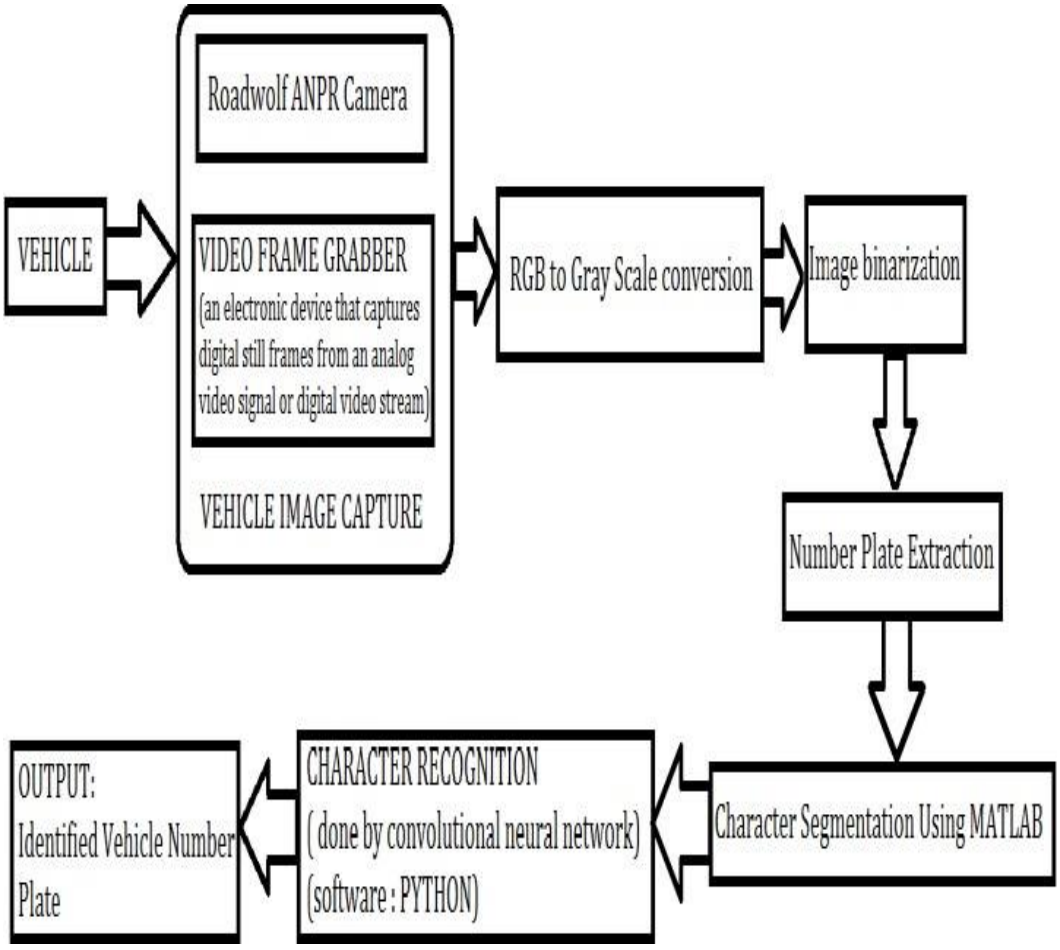


Figure 3.2-Block diagram of ANPR System

ACQUIRED INPUT IMAGE

The first step is to acquire the image of the vehicle with license plate which in our case is taken from an already existing set of images been acquired with the help of various devices (viz. high speed cameras etc.) as shown in Figure 3.2

IMAGE PREPOCESSING

Image pre-processing is an important step in any image analyzing system. Without a proper pre-processing, the recognition will be ineffective or may give improper results in later stages. The main motive of pre-processing is to enhance the quality of the image that will be processed for recognition.

Various processes that we are going to apply are converting RGB image to Grayscale, noise reduction and binarization of image.

RGB to Gray Scale Conversion

The inputed image is in RGB format. Main purpose of this conversion is to reduce the number of colors as shown in Figure 3.3

Figure 3.3 -Input Image

Figure 3.4-Conversion of Grayscale

NOISE REDUCTION

Image noises are distortion in the image that arises due to fault in camera or result of poor visibility due to changing weather conditions. Noises are also the random variation in the intensity levels of the pixels. Noise can be of various types like Gaussian noise, Salt and pepper noise. In this proposed method, we use iterative bilateral filter for noise removal. It provides the mechanism for noise reduction while

preserving edges more effectively than median filter. Figure 3.5 and3.6. shows the images with noise and reduced noise respectively.

Figure 3.5-Image with reduced noise

Figure 3.6-Image with noise

BINARIZATION

Binarization is the process of converting an image into an image with two pixels value only i.e. containing white and black pixels. Performing binarization process before detecting and extracting license plate from the image will make the task of detecting license plate easier as edges will be more clearly in binary image.

Binarization is performed by selecting a threshold value. After selecting the value, we analyze the pixel values in the image. If it's greater than threshold, then make that pixel fully white or black accordingly. This is simple thresholding method which may not yield proper result by selecting a global threshold value. Hence to overcome this, we use an adaptive thresholding method in which instead of selecting a global threshold value we calculate threshold of smaller region in the image which gives better result.

LICENSE PLATE EXTRACTION

In license plate extraction, it is important to take in mind the boundaries of the plate in the image as in [10]. For doing so, we have many methods such as Sobel's edge detection method and Hough's Line detection method. Now the connected component method helps us to find out what the shape actually is by the grouping method of intersection points of the shapes

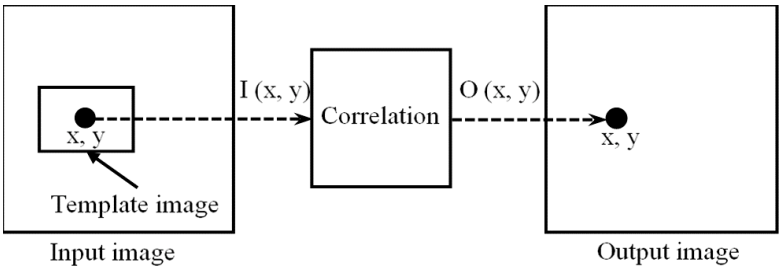


Figure 3.7- Template matching technique for detecting the vehicle number plate.

Further by getting the intersection points of the shapes, we come to known whether it is a rectangle or not depending upon the number of points in that respective group. Since now we have got the points of rectangles, we can successfully extract the rectangular parts from the image out of which we can get the license plate depending upon some properties of the license plate such as major axis length, minor axis length, area, bounding box etc. as shown in Figure 3.7**.**

Now the extracted plate is actually an inverted binary image of it from the actual image of the car as shown in Figure **3.8** .Further steps cannot be applied on such an image. Thus we convert the image into binary image for further operations as shown in Figure **3.09**

Figure 3.8-Extracted license plate

Figure 3.9-Inverted binary image of LP

Figure 3.10-Binary image of LP

CHARACTER SEGMENTATION

Character segmentation is done on the binary image of the extracted license plate. The algorithm used for the same is horizontal scanning which makes use of a scanning line which finds the conditions satisfying the start and end position of the character as in [11]. The pseudo code for the same is given below in Figure **3.11**

```
global var start_pos
global var end_pos

function start_of_char(img)
{
increment the scan_line
start_pos=position(scan_line)
until(finds(any_one_pixel(scan_line)),1)==true)
else goto function end_of_char(img)
}

function end_of_char(img)
{
increment the scan_line
end_pos=position(scan_line)
until(finds(all_pixel(scan_line),0)==true)
else exit
}
```

Figure 3.11 -Pseudo-code of character segmentation

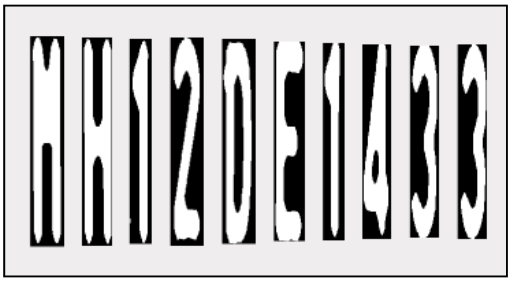
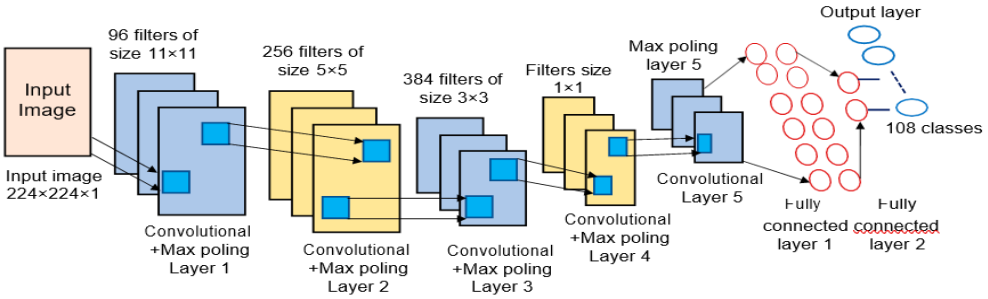


Figure 3.12-Segmented characters

CHARACTER RECOGNITION USING CNN

In order to recognize the segmented characters efficiently, we used artificial neural network training to train our system over a dataset downloaded from [12]. After this training, we used the same neural model to recognize the characters.

We used a CNN with 2 convolution layers at the beginning and 2 fully connected layers at the end. We used a dataset to train the CNN as referred from [15][16][17]. The dataset consists of 1,000 sample images for each of the 36 characters. Out of the 36,000 samples, we used the first 30,000 samples as training data and remaining 6,000 samples as test data. We first trained the model with the number of training steps being 100 followed by testing.



The CNN has 784 input nodes. The first convolution layer has 5x5 kernels followed by a pooling layer of size 2x2. Output layer has 36 nodes. We used tensorflow to train and test our model. We used gradient descent algorithm to minimize cross-entropy with having a learning rate of 0.5.

3.2.2 Objectives of Proposed System

The following are the objectives of the proposed system in the project.

- Maintenance cost reduction.
- Effective.
- Affordable.
- Reduction of manpower.
- Human Error avoidance
- Secure
- Low cost
- Product excellence due to minimum requirements

- Optimized solution
- Increased reliability and faster reaction time compared to human workers

These are the objectives of the proposed system in the project "Automatic Vehicle Number Plate Detection Using Neural Networks".

3.2.3 Advantages of Proposed system

It is definitely a better alternative to the existing proprietary systems, even though there are known restrictions with high resolution to detect the plate using OpenCV and python which is most easy to understand and make changes.

3.3 DEFINATIONS

- **Machine Learning**

It is the process of organizing, analyzing and modeling data. Regression and classification is used for predictive modeling .In simple words training them a chine to learn. First of we need some data to train the machine after this there is a need to cross validate the learning and at the end we need to test either the results are near to the requirement or not. If the results are near to the requirements it means that our machine is learning well otherwise we need to train the machine again and cross validate again and this process will repeat till our requirement meet.

For learning machine there are number or techniques and many algorithms that we can use. Here are two types of machine learning techniques:

1. Supervised machine learning
2. Unsupervised machine learning

1. Supervised machine learning

Supervised learning (machine learning) takes a known set of input data and known responses to the data, and seeks to build a predicter model that generate reasonable predictions for the response to new data.

Suppose you want to predict if someone will have a heart attack within a year.

You have a set of data on previous people, including age, weight, height, blood pressure, etc. You know if the previous people had heart attacks within a year of their data measurements. So the problem is combining all the existing data into a model that can predict whether a new person will have a heart attack within a year.
Supervised learning splits into two broad categories:

1. **Classification** for responses that can have just a few known values, such as 'true' or 'false'. Classification algorithms apply to nominal, not ordinal response values.
2. **Regression** for responses that are a real number, such as miles per gallon for a particular car.

Few Supervised classification learning techniques are

- Logistic regression
- Generalized linear model
- Classification
- Support vector machine
- Artificial neural network

➢ **Artifical Neural Networks**

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the connections between elements largely determine the network function. You can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

Typically, neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Typically, many input/target pairs are needed to train a network.

Neural networks have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, speech, vision, and control systems.
Artificial neural network is same as human nervous system. In a neural there are three

types of layers:

1. Input layer
2. Hidden layers
3. Output layer

Input layer contains input data that we need to feed the neural network. Number of inputs depend on the data samples.

Hidden layers contains some hidden nodes and bias nodes. Number of nodes depends on user and complexity of network and can be increased accordingly.
Not just nodes but hidden layers can also be increased as per requirements. This network will call multilayer neural network.

Output layer contain output nodes. Number nodes depend on the classes of data number of classes and output nodes will remain same [2].

- **Neural Network Design Steps**

We will follow the standard steps for designing neural networks to solve problems in four application areas: function fitting, pattern recognition, clustering, and time series analysis. The work flow for any of these problems has seven primary steps [4].

1. Collect data
2. Create the network
3. Configure the network
4. Initialize the weights and biases
5. Train the network
6. Validate the network
7. Use the network

- **Topologies of Neural network**

Here are some Topologies / Architectures and algorithm that can be used for Artificial Neural Networks ANN:

- Feed forward neural network
- Recurrent neural network
- Multi-layer perceptron(MLP)

- Convolutional neural networks
- Recursive neural networks
- Deep belief networks
- Self-Organizing Maps
- Deep Boltzmann machine
- Stacked de-noising auto-encoders

Non-linear / Logistic regression technique in which Feed forward network algorithm is used is followed here.

➢ **Logistic Regression**

Logistic regression is the type of predictive model that can be used when the target variable with two categories–for example live/die, has disease/ doesn't have disease, purchase product/ doesn't purchase product, etc. A logistic regression model doesn't involve decision trees and more similar to Non-linear regression such as fitting a polynomial to a set of data values. Logistic regression can be used only with two types of target variables:[5]

- A **Categorical target variable** that has exactly two categories (i.e. a binary variable).
- A **Continuous target variable** that has values in the range 0.0 to 1.0 representing probability values or proportions.

➢ **Logistic Function**

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits [5],[6].

➢ **Feedforward networks**

Feedforward networks consist of a series of layers. The first layer has a

connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer produces the network's output.

Feedforward networks can be used for any kind of input to output mapping. A feedforward network with one hidden layer and enough neurons in the hidden layers, can fit any finite input-output mapping problem.

- **Back propogation**

Back propagation is a method used in artificial neural networks to calculate a gradient that is needed in the calculation of the weights to be used in the network. It is commonly used to train deep neural networks, a term referring to neural networks with more than one hidden layers.

Back propagation is a special case of an older and more general technique called automatic differentiation. In the context of learning, back propagation is commonly used by the gradient descent optimization algorithm to adjust the weight of neurons by calculating the gradient of the loss function. This technique is also sometimes called backward propagation of errors, because the error is calculated at the output and distributed back through the network layers. The back propagation algorithm has been repeatedly rediscovered and is equivalent to automatic differentiation in reverse accumulation

Mode . Back propagation requires a known, desired output for each input value to the weights of the network. This is normally done using back propagation. Assuming one output neuron, the squared error function is:

$$E=^{1}(t-y)^{2},\ \text{also called 'mean squared error'}$$

Where E is the squared error, t is the target output for a training sample, and y is the actual output of the output neuron.

- **Evaluation of supervised Classification Algorithms**

In binary classification here are two possible output classes(0 or 1,yes or no, true or false, positive or negative etc.)

- What measures should weuse?

- Classification accuracy might not beenough.
- How reliable are the predictedvalues?
- Are errors on the training data a good indicator of perform ance on future data?
- If the classifier is computed from the very same training data, any estimate based on that data will be optimistic.
- New data probably not exactly the same as the training data[7].

➢ Confusion Matrix

The Confusion matrix is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It is used for Classification problem where the output can be of two or more types of classes [7][8]

Predicted class **Actual class**	**Yes**	**No**
Yes	True positives=100	False negatives=5
No	False positives=10	True negatives=50

Table 3.1-Confusion Matrix

N=165, Actual yes = 105, Actual no = 60 predicted yes = 110, predicted no = 55.

True positives: positive tuples correctly labeled**.**
True negatives: negative tuples correctly labeled**.**
False positives: negative tuples incorrectly labeled.
False negatives: positive tuples incorrectly labeled.

➢ Case Study of confusion matrix

Suppose we have a two-class classification problem of predicting whether a

photograph contains a man or woman.

We have a test data of 10 records with expected outcomes and a set of predictions from our classification algorithm which is shown in Table 3.1

No.	**Expected outcome**	**Predicted**
1	1	0
2	1	1
3	0	0
4	1	1
5	0	1
6	0	0
7	0	0
8	1	1

Table 2.2-Case study of Confusion Matrix

Let’s turn our results into a confusion matrix.

First we must calculate the number of correct predictions for each class as shown in Table 3.1

1	1	3
2	0	4

We can calculate number of incorrect predictions for each class organized by the predicted value as shown in below Table

1	1	2
2	0	1

- **Class Confusion matrix**

Actual class	Predicted	
	1	0
Men	3	1
Women	2	4

Table 3.3-shows 2-class confusion matrix

3.4 CONVOLUTIONAL NETWORKS DESIGN

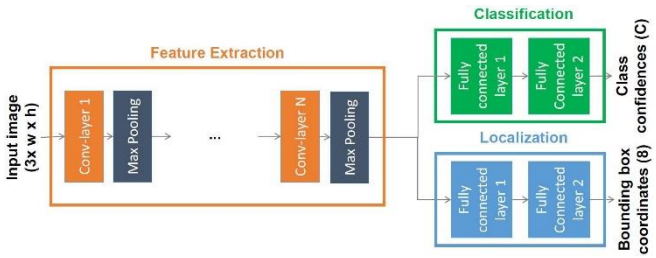


Figure 3.13 - General CNN architecture for object detection.

In this section we first describe our CNN for generic object detection, where object detection consists in the two sub-tasks of object classification and localization. Then, we detail how to adjust the network hyper parameters to turn it into a plate or a character detector.

General Architecture

The network receives as input a fixed size RGB image that is fed to a feature extraction stage composed by a cascade of convolutional blocks. Each convolutional block includes

1. One convolutional layer with 3×3 filters and Rectified Linear Units(ReLU) activation functions
2. Max-pooling layers for feature spatial down sampling [4]. The extracted features are forwarded to the object classifier and the object localizer, as features useful for object classification are expected to be useful also for object localization and the other way around.

The classifier stage is a fully connected network predicting if the input image contains an object belonging to the i^{th} class out of C possible classes. This stage has C outputs and includes a *soft max layer* yields normalized *confidence ratings* for each class. The localizer stage is a linear regressor and predicts the position of the object within the image (namely, the surrounding bounding box vertices). This stage has 8 outputs for the 4 (x,y) coordinate pairs describing the object bounding boxvertices.

We now discuss how to adapt the above CNN to the distinct applications of plate detection and character detection.

Plate Detector Architecture

The term plate detection here indicates the joint tasks of i) predicting if a plate is present in the input image (classification) and ii) localizing the 4 corners of the plate. We consider 128×64 input images to accommodate for the aspect ratio of EU-format plates. The plate detector feature extraction stage consists of 10 3×3 convolutional layers with ReLUs and with 4 Max poolings after the 2^{nd}, 4^{th}, 7^{th} and 10^{th} convolutional layer. The classifier stage consists of 3 fully connected layers and has C=2 outputs, corresponding to the confidence that a plate is present or not in the input image. The localization stage consists of 3 fully connected layers connected to the 8 regression units defining the bounding box surrounding the plate.

The plate detector has 10 convolutional layers in place of 8~16 layers of [4] as the input image size is smaller (128×64 vs. 224×224). Furthermore, the number of convolutional filters is equal to 16, 32, 64, 128 rather than 64, 128, 256, 512 respectively. The dimensions of the plate classification problem are in fact intrinsically different than in [3][4] (C=2 in place of C=1000). Thus, as fewer features have to be learned, we reduce the number of trainable parameters reducing the training complexity (our experiments showed little performance gains when increasing the filter sizes).

Character Detector Architecture

Similarly, the term character detection here indicates the joint tasks of i) identifying characters within an image assigning a class confidence to each of them and ii) localizing the character bounding box corners. The character detector CNN architecture is similar to the plate detector with some key differences. The input image is smaller, 24×40 pixels, suitable to account for the wide range of characters used in EU plates. The feature extraction stage consists of 7 3×3 convolutional layers with ReLUs and 3 Max Pooling after the 2^{nd}, 4^{th} and 7^{th} convolutional layer.

The classifier stage has 3 fully connected layers and C=33 outputs, one for each of the 32 eligible characters in Italian plates plus one output for the case where no character is detected. The localization stage consists of 3 separated fully connected layers connected to the 4 regression units defining the bounding box surrounding the plate.

The character detector feature extraction stage has 7 convolutional layers, fewer than the plate detector, due to the smaller input image size (24×40 vs 128×64). The number of filters is 32, 64, 128, 256; larger than the plate detector but lower than [4]. In fact, the character detector network needs to learn more features to correctly differentiate images belonging to C=33 classes in part similar to each other (e.g., characters 7, T and1).

Finally, the CNNs may be converted to fully convolutional for efficient multiple, arbitrary-sized, object detection at arbitrary positions in the image (e.g., plates typically appear with arbitrary size in an image and multiple characters are typically found in a plate) [3]. The first fully connected layer in the classifier and in the localizer are

rearranged into a convolutional layer with a filter size equal to the dimensions of the output from the last feature extraction stage (8×4 for the plate detector, 3×5 for the character detector). Next, the following two fully connected layers are converted to 1×1 convolution. So, the plate and character detection networks are capable of object detection within a number of overlapping input windows in the input image of size 128×64 and 24×40 respectively. Table 1 shows that the number of trainable parameters of our CNNs is one order of magnitude lower than [3][4].

Number of parameters	OverFeat (accurate model)	VGGNet(conf.C)	Proposed (plate detection)	Proposed (character detection)
Convolutional layers	~ 19M (6)	~ 10M (13)	478k (10)	434k (7)
Fully connected layers	~ 125M (3)	~ 124M (3)	~ 9M (3+3)	~12M (3+3)
All layers	~ 144M (9)	~ 134M (16)	~ 10M (16)	~12M (13)

Table 3.4- Number of trainable parameters and layers in parenthesis

3.5 DATASET GENERATION AND TRAINING PROCEDURES

In this section, we first describe the procedure to generate synthetic car plate images. Then, we describe how to extract synthetic samples suitable for training the plate and the character detector CNNs introduced in the previous section. Finally, we detail the CNN training procedure:

Dataset Images

Synthetic Images Generation

First, we generate synthetic images containing either one plate (*positive* images) or no plates at all (*negative* images) .\

Positive images are 1024×768 pixels in size and contain one synthetic plate in the center of the image. First, we generate a synthetic 2D plate template matching the EU EC

2411/98 recommendation. Then, we superimpose a random 7- characters string matching the IT numbering scheme (2 letters, 3 digits, 2 letters) using the appropriate font and spacing. We account for the fact that real plates are embossed by adding random reflections and shadows around the characters and adjusting the font thickness. Next, we randomly add different levels of shading to simulate partial or total plate occlusions by other car parts (e.g., plates spotted from narrow angles).

The plate is then projected over a different 2D surface to simulate camera capture from a random angle. The image is then scaled so that the plate width ranges between 80 and 200 pixels, such that the smallest plates are still readable by naked eye while the largest are at least twice as large as the smallest. Since a camera capture will alter the colors (due to e.g. light conditions) the image is converted to HSL-color space and each channel is slightly altered at random. The plate is finally superimposed on a random natural background image with variable transparency factor.

Negative images are equally sized and contain either random natural background or a randomly generated text string with random font and color over the random background to make the plate detector robust to false positives (e.g., spurious text cluttering the scene).

Positive and negative images may be randomly altered adding some blur to simulate motion and noise to simulate scarce illumination conditions.
Notice that by generating synthetic training images, we overcome the problem of acquiring a sufficient number of fully-annotated samples (plate readings and exact position of each plate and character bounding box) for training our CNNs.

Plate Samples Selection

The plate detector is trained on 128×64 samples extracted at random from the scaled synthetic images to avoid over fitting. Positive samples are cropped at random so that each sample contains the entire plate. Negative samples are either random crops from negative images or random crops of positive images such that less than 50% of the plate is present in the sample. Such approach makes the plate detector robust to partial plate

views, reducing the expected number of false positive detections. We eventually generate 25,000 positive and 25,000 negative samples to train the plate detector.

Character Sample Selection

The character detector is trained on 24×40 samples extracted at random from the synthetic images. Positive samples consist in random crops of the synthetic images so that each crop contains one character fitting entirely into the 24×40 character detector input window size. Moreover, crops are randomly rescaled so that the actual character height ranges between 28 and 34 pixels to account for slight character size variations due to perspective changes. Negative samples are either random crops from negative images that do not contain characters at all, or positive images crops such that less than 50% of a character is contained, making the character detector robust to partial character views. Finally, the character detector training set consists of about 175,000 positive and 175,000 negative training samples.

Training Procedures

The plate and the character detectors are separately trained for 100 epochs each with random parameter initialization and back propagation of the learned parameters. The minimized loss function is the Mean Square Error (MSE) over all localization and classification outputs. We consider batches of 128 samples and at the end of each batch we move towards the averaged steepest gradient direction. The parameters update process is optimized via Adagrad[9], where the initial learning rate value is equal to 5×10^{-2} and the learning rate decay factor is equal to 10^{-3}.

3.6 SYSTEM ALGORITHM

LPs are located by means of horizontal and vertical projections through search window in [1]. If the LP is not found using the above method then the original image is inverted. Since LPs are located at the bottom part of the image, projection histograms are scanned from the bottom to the top so that the height of the LP can easily be identified. Figure4.1, shows the result of horizontal segmentation. After performing the horizontal

segmentation, vertical projection is carried out. Figure 4.2, shows the result of vertical segmentation. In [2], region with dense vertical edges is segmented as a candidate plate which is known as ROI. In the location procedure, vertical Sobel edge features are primarily extracted. Then a skeleton extraction algorithm on edge map is performed. There is a possibility that dense pixels are text region and isolated edge pixels are often noises. So density based region growing method is used to locate candidate LP regions. Figure 3.14 shows the ROI location.

Figure 3.14-Horizontal segmentation

Figure 3.15-verstical segmentation

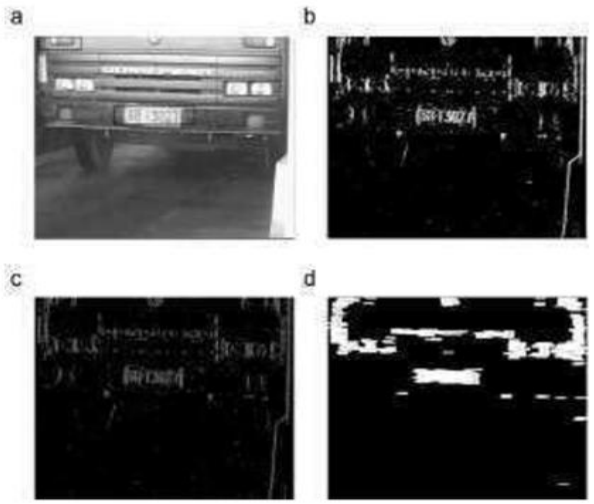
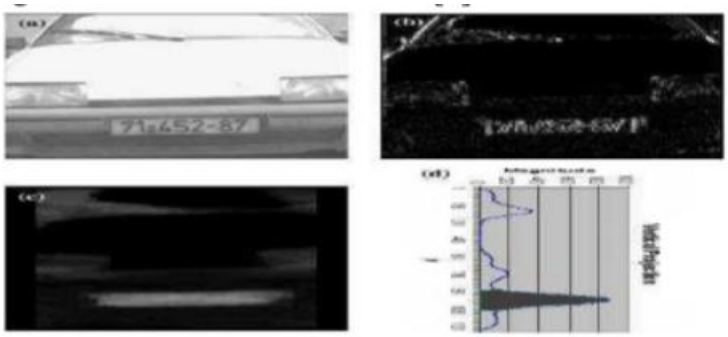


Figure 3.16-Illustration of ROI location

Connected Component Labelling (CCL) is used for LPD. CCL scans the image and labels the pixels according to the pixel connectivity. There are two types of connectivity:-4 and 8 connectivity. In a feature extraction algorithm is used to count the similar labels to distinguish it as a region. The region with maximum area is considered as a possible license plate region and this region is forwarded to the segmentation process. But in [10], two detection methods are performed. One is detection of white frame and another one is detection of black characters. White frame is detected using CCL technique and it is sensitive to the edges. So if the frame is broken, the LP cannot be located properly. To determine the candidate frames, aspect ratio of the LP, height and width of characters have to be known. Further, the penetration times through the midline of the large numerals in the LP is also calculated for candidate frame selection. In [4], after the successful CCL on the binary image, measurements such as orientation, aspect ratio, Euler number for every binary object in the image are calculated. Criteria such as orientation < 35 degrees, 2 < aspect ratio < 6 and Euler number > 3 are considered as candidate plate regions in [4]. But this method does not guarantee that LP with dark background and white characters will be successfully detected. In [5], fuzzy logic is used to locate LP. The author framed some rules to explain about the LP and gave some membership functions for fuzzy sets – "bright","dark","bright and dark sequence","texture" and "yellowness" to obtain the horizontal and vertical plate positions. But this need is very sensitive to the LP color and brightness. It also takes longer processing time compared to conventional color based methods. In [6], vertical edge detection method is utilized for locating the LP. So Robert's edge detector is used to

emphasize the vertical edges. There will be many abrupt intensity changes but a cluster of 10-15 sharp intensity changes is considered as plate zone. Image is convolved with horizontally oriented rank-filter of M×N pixels. This leads to a bright-elongated spot of ellipsoidal shape in the plate's area. The last step is horizontal projection. Figure 3.16 illustrates the LPD used in [6].



In [11], after applying the Sobel edge detection method certain rules have been applied to locate the LP area. The first criterion is to find the column range of LP. Second step is to detect row range. After these steps, license plate will be obtained. So, to eliminate some candidate, the pseudo license plate aspect ratio of each candidate region is calculated. In [9], a combination of Hough transform and counter algorithm is applied to

detect the LP region. First the counter algorithm is applied to detect the closed boundaries of objects. These counter lines are transformed to Hough coordinate to find interacted parallel lines that are considered as LP candidates. Figure 3.19 shows the result of using the counter and Hough transform method. To filter out the candidate plates, the aspect ratio of the LP and the horizontal cross cuts are used. In the case of number of horizontal cross cuts for 1 row plate is in the range of 4 to 8. For 2 row plate, it is in the range of 7 to 16.

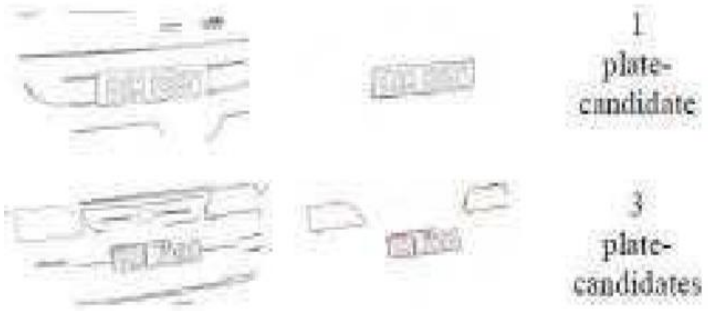


Figure 3.17-Result of combination of Counter and Hough transform method

3.7 SOFTWARE MODULES

1. Python Programming Language
2. OpenCV
3. Numpy
4. Scikit-Learn
5. Matplotlib
6. Keras

3.7.1 PYTHON PROGRAMMING LANGUAGE

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming onboth small and large scales. Van Rossum led the language community until stepping down as leader in July 2018.Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural.It also has a comprehensive standard library. The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third-party Python modules, programs and tools, and additional documentation.

Python is simple to use, but it is a real programming language, offering much more structure and support for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much more error checking than C, and, being a very-high-level language, it has high-level data types built in, such as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those languages.

Python allows you to split your program into modules that can be reused in other Python programs. It comes with a large collection of standard modules that you can use as the basis of your programs — or as examples to start learning to program in Python. Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like Tk.

Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throw-away programs, or to test functions during bottom-up program development. It is also a handy desk calculator.

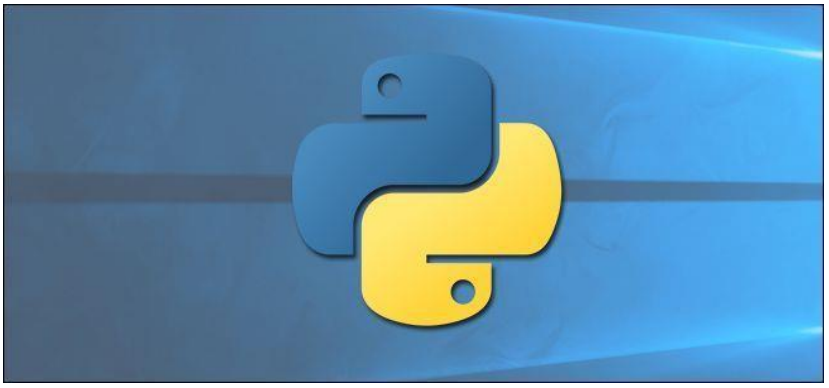
Python enables programs to be written compactly and readably. Programs written in Python are typically much shorter than equivalent C, C++, or Java programs, for several reasons:

- the high-level data types allow you to express complex operations in a single statement;
- statement grouping is done by indentation instead of beginning and ending brackets;

- no variable or argument declarations are necessary.

Python is extensible : if you know how to program in C it is easy to add a new built-in function or module to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to libraries that may only be available in binary form (such as a vendor-specific graphics library). Once you are really hooked, you can link the Python interpreter into an application written in C and use it as an extension or command language for that application.

How to Install Python on Windows



Python doesn't come prepackaged with Windows, but that doesn't mean Windows users won't find the flexible programming language useful. It's not quite a simple as installing the newest version however, so let's make sure you get the right tools for the task at hand.

First released in 1991, Python is a popular high-level programming language used for general purpose programming. Thanks to a design philosophy that emphasizes readability it has long been a favorite of hobby coders and serious programmers alike. Not only it os an easy language (comparatively speaking, tha tis) to pickup but you'll find thousands of projects online that require you have Python installed on set the program.

Which Version Do You Need?

Unfortunately, there was a significant update to Python several years ago that

created a big split between Python versions. This can make things a bit confusing to newcomers, but don't worry. We'll walk you through installing both major versions.

When you visit the Python for Windows download page, you'll immediately see the division. Right at the top, square and center, the repository asks if you want the latest release of Python 2 or Python 3 (2.7.13 and 3.6.1, respectively, as of this tutorial).

3.7.2 OPENCV

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- Core functionality (core) - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- Image Processing (imgproc) - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- Video Analysis (video) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.

- Camera Calibration and 3D Reconstruction (calib3d) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- 2D Features Framework (features2d) - salient feature detectors, descriptors, and descriptor matchers.
- Object Detection (objdetect) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- High-level GUI (highgui) - an easy-to-use interface to simple UI capabilities.
- Video I/O (videoio) - an easy-to-use interface to video capturing and video codecs.
- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

OpenCV-Python

Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.

Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in

background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation.

And the support of Numpy makes the task more easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this.

So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

Automatic Memory Management

OpenCV handles all the memory automatically.

First of all, std::vector, cv::Mat, and other data structures used by the functions and methods have destructors that deallocate the underlying memory buffers when

needed. This means that the destructors do not always deallocate the buffers as in case of Mat. They take into account possible data sharing. A destructor decrements the reference counter associated with the matrix data buffer. The buffer is deallocated if and only if the reference counter reaches zero, that is, when no other structures refer to the same buffer. Similarly, when a Mat instance is copied, no actual data is really copied. Instead, the reference counter is incremented to memorize that there is another owner of the same data. There is also the Mat::clone method that creates a full copy of the matrix data.

Automatic Allocation of the Output Data

OpenCV deallocates the memory automatically, as well as automatically allocates the memory for output function parameters most of the time. So, if a function has one or more input arrays (cv::Mat instances) and some output arrays, the output arrays are automatically allocated or reallocated. The size and type of the output arrays are determined from the size and type of input arrays. If needed, the functions take extra parameters that help to figure out the output array properties.

The array frame is automatically allocated by the >> operator since the video frame resolution and the bit-depth is known to the video capturing module. The array edges is automatically allocated by the cvtColor function. It has the same size and the bit-depth as the input array. The number of channels is 1 because the color conversion code cv::COLOR_BGR2GRAY is passed, which means a color to grayscale conversion. Note that frame and edges are allocated only once during the first execution of the loop body since all the next video frames have the same resolution. If you somehow change the video resolution, the arrays are automatically reallocated.

The key component of this technology is the Mat::create method. It takes the desired array size and type. If the array already has the specified size and type, the method does nothing. Otherwise, it releases the previously allocated data, if any (this part involves decrementing the reference counter and comparing it with zero), and then allocates a new buffer of the required size. Most functions call the Mat::create method for each output array, and so the automatic output data allocation is implemented.

Some notable exceptions from this scheme are cv::mixChannels, cv::RNG::fill, and a few other functions and methods. They are not able to allocate the output array, so you have to do this in advance.

Saturation Arithmetics

As a computer vision library, OpenCV deals a lot with image pixels that are often encoded in a compact, 8- or 16-bit per channel, form and thus have a limited value range. Furthermore, certain operations on images, like color space conversions, brightness/contrast adjustments, sharpening, complex interpolation (bi-cubic, Lanczos) can produce values out of the available range. If you just store the lowest 8 (16) bits of the result, this results in visual artifacts and may affect a further image analysis. To solve this problem, the so-called saturation arithmetics is used. For example, to store r, the result of an operation, to an 8-bit image, you find the nearest value within the 0..255 range:

I(x,y)=min(max(round(r),0),255)

Similar rules are applied to 8-bit signed, 16-bit signed and unsigned types. This semantics is used everywhere in the library. In C++ code, it is done using the cv::saturate_cast<> functions that resemble standard C++ cast operations.

Fixed Pixel Types. Limited Use of Templates

Templates is a great feature of C++ that enables implementation of very powerful, efficient and yet safe data structures and algorithms. However, the extensive use of templates may dramatically increase compilation time and code size. Besides, it is difficult to separate an interface and implementation when templates are used exclusively. This could be fine for basic algorithms but not good for computer vision libraries where a single algorithm may span thousands lines of code. Because of this and also to simplify development of bindings for other languages, like Python, Java, Matlab that do not have templates at all or have limited template capabilities, the current OpenCV implementation is based on polymorphism and runtime dispatching over templates. In those places where runtime dispatching would be too slow (like pixel access operators), impossible (generic cv::Ptr<> implementation),or just very inconvenient (cv::saturate_cast<>()) the current implementation introduces small template classes, methods, and functions. Anywhere else in the current OpenCV version the use of templates is limited.

Consequently, there is a limited fixed set of primitive data types the library can operate on. That is, array elements should have one of the following types:

- 8-bit unsigned integer (uchar)
- 8-bit signed integer (schar)
- 16-bit unsigned integer (ushort)
- 16-bit signed integer (short)
- 32-bit signed integer (int)
- 32-bit floating-point number (float)
- 64-bit floating-point number (double)

A tuple of several elements where all elements have the same type (one of the above). An array whose elements are such tuples, are called multi-channel arrays, as opposite to the single-channel arrays, whose elements are scalar values. The maximum possible number of channels is defined by the CV_CN_MAX constant, which is currently set to 512.

InputArray and OutputArray

Many OpenCV functions process dense 2-dimensional or multi-dimensional numerical arrays. Usually, such functions take cppMat as parameters, but in some cases it's more convenient to use std::vector<> (for a point set, for example) or cv::Matx<> (for 3x3 homography matrix and such). To avoid many duplicates in the API, special "proxy" classes have been introduced. The base "proxy" class is cv::InputArray. It is used for passing read-only arrays on a function input. The derived from InputArray class cv::OutputArray is used to specify an output array for a function. Normally, you should not care of those intermediate types (and you should not declare variables of those types explicitly) - it will all just work automatically. You can assume that instead of InputArray/OutputArray

You can always use Mat, std::vector<>, cv::Matx<>, cv::Vec<> or cv::Scalar. When a function has an optional input or output array, and you do not have or do not want one, pass cv::noArray().

Error Handling

OpenCV uses exceptions to signal critical errors. When the input data has a correct format and belongs to the specified value range, but the algorithm cannot succeed for some reason (for example, the optimization algorithm did not converge), it returns a special error code (typically, just a boolean variable).

The exceptions can be instances of the cv::Exception class or its derivatives. In its turn, cv::Exception is a derivative of std::exception. So it can be gracefully handled in the code using other standard C++ library components.

Multi-threading and Re-enterability

The current OpenCV implementation is fully re-enterable. That is, the same function or the same methods of different class instances can be called from different threads. Also, the same Mat can be used in different threads because the reference-counting operations use the architecture-specific atomic instructions.

3.7.3 NUMPY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific and engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package. Among its members was Python designer and maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier.

An implementation of a matrix package was completed by Jim Fulton, then generalized by Jim Hugunin to become Numeric,[4] also variously called Numerical Python extensions or NumPy. Hugunin, a graduate student at Massachusetts Institute of Technology (MIT), joined the Corporation for National Research Initiatives (CNRI) to work on JPython in 1997 leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to take over as maintainer. Other early contributors include David Ascher, Konrad Hinsen and Travis Oliphant.

A new package called Numarray was written as a more flexible replacement for Numeric. Like Numeric, it is now deprecated. Numarray had faster operations for large arrays, but was slower than Numeric on small ones, so for a time both packages were used for different use cases. The last version of Numeric v24.2 was released on 11 November 2005 and numarray v1.5.2 was released on 24 August 2006.

There was a desire to get Numeric into the Python standard library, but Guido van Rossum decided that the code was not maintainable in its state then.

In early 2005, NumPy developer Travis Oliphant wanted to unify the community around a single array package and ported Numarray's features to Numeric, releasing the result as NumPy 1.0 in 2006.[7] This new project was part of SciPy. To avoid installing the large SciPy package just to get an array object, this new package was separated and called NumPy. Support for Python 3 was added in 2011 with NumPy version 1.5.0.[13]

In 2011, PyPy started development on an implementation of the NumPy API for PyPy. It is not yet fully compatible with NumPy.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing byte code interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted,[16] and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

Traits

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted,[16] and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

The ndarray data structure

The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure. These arrays are strided views on memory.[7] In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type.

Such arrays can also be views into memory buffers allocated by C/C++, Cython, and Fortran extensions to the CPython interpreter without the need to copy data around, giving a degree of compatibility with existing numerical libraries. This functionality is

exploited by the SciPy package, which wraps a number of such libraries (notably BLAS and LAPACK).NumPy has built-in support for memory mapped ndarrays.[7]

Limitations

Inserting or appending entries to an array is not as trivially possible as it is with Python's lists. The np.pad(...) routine to extend arrays actually creates new arrays of the desired shape and padding values, copies the given array into the new one and returns it. NumPy's np.concatenate([a1,a2]) operation does not actually link the two arrays but returns a new one, filled with the entries from both given arrays in sequence. Reshaping the dimensionality of an array with np.reshape(...) is only possible as long as the number of elements in the array does not change. These circumstances originate from the fact that NumPy's arrays must be views on contiguous memory buffers. A replacement package called Blaze attempts to overcome this limitation.[17]

Algorithms that are not expressible as a vectorized operation will typically run slowly because they must be implemented in "pure Python", while vectorization may increase memory complexity of some operations from constant to linear, because temporary arrays must be created that are as large as the inputs. Runtime compilation of numerical code has been implemented by several groups to avoid these problems; open source solutions that interoperate with NumPy include `scipy.weave` ,numexpr[18] and Numba.[19] Cython and Pythran are static-compiling alternatives to these.

3.7.4 SCIKIT-LEARN

Defining scikit learn, it is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k- means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-learn was initially developed by David Cournapeau as a Google summer of code project in 2007.Later Matthieu Brucher joined the project and started to use it as a part of his thesis work. In 2010 INRIA got involved and the first

public release (v0.1 beta) was published in late January 2010.The project now has more than 30 active contributors and has had paid sponsorship from INRIA, Google, Tinyclues and the Python Software Foundation.

In general, a learning problem considers a set of n samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and,for instance, a multi-dimensional entry (aka multivariate data), it is said to have several attributes or features.

Learning problems fall into a few categories:

- supervised learning,in which the data comes with additional attributes that we want to predict (the scikit-learn supervised learning page).This problem can be either:
- classification : samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data. An example of a classification problem would be handwritten digit recognition, in which the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the n samples provided, one is to try to label them with the correct category or class.

- regression: if the desired output consists of one or more continuous variables, then the task is called *regression*. An example of a regression problem would be the prediction of the length of a salmon as a function of its age andweight.

- unsupervised learning, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of *visualization* (Scikit-Learn unsupervised learning page).

Training set and testing set

Machine learning is about learning some properties of a data set and then testing those properties against another data set. A common practice in machine learning is to evaluate an algorithm by splitting a data set into two. We call one of those sets the **trainingset**, on which we learn some properties ;we call the other set the **testingset**, on which we test the learned properties.

Loading an example dataset

scikit-learn comes with a few standard datasets, for instance the iris and digits datasets for classification and the boston house prices dataset for regression.

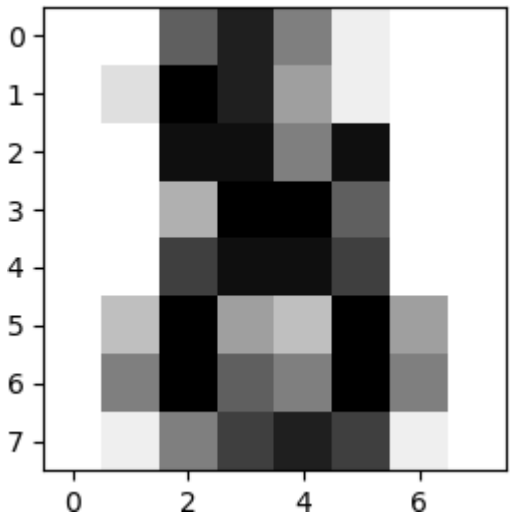
Shape of the data arrays

The data is always a 2D array, shape (n_samples, n_features), although the original data
may have had a different shape. In the case of the digits, each original sample is an image of shape (8, 8) and can be accessed using:

```
>>
>
```

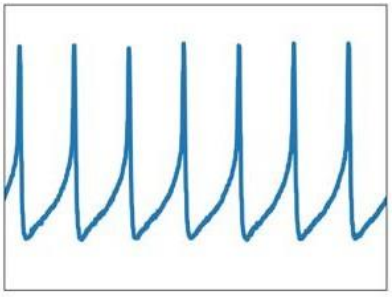
In the following, we start a Python interpreter from our shell and then load the irisand digits datasets. Our notational convention is that $ denotes the shell prompt while >>> denotes the Python interpreter prompt:

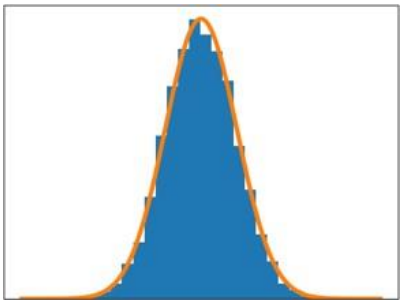
```
$ python
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> digits = datasets.load_digits()
```

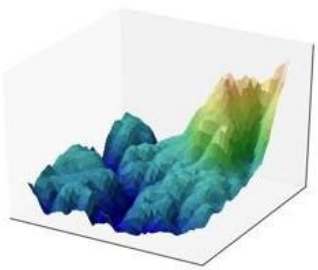


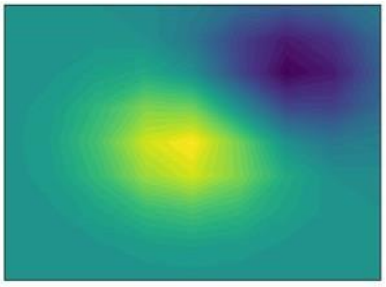
3.7.5 MATPLOTLIB

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook ,web application servers ,and four graphical user interface toolkits.









Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties,axes properties,etc,via an object oriented interface or via a set of functions familiar to MATLAB users.

Installing an official release

Matplotlib and its dependencies are available as wheel packages for macOS, Windows and Linux distributions:

```
python -m pip
install -U pip
python -m pip
```

Although not required, we suggest also installing IPythonfor interactive use. To easily install a complete Scientific Python stack, see Scientific Python Distributions below.

- **macOS**

To use the native OSX backend you will need a framework build build of Python.

- **Testdata**

The wheels (*.whl) on the PyPI download page do not contain test data or example code.

If you want to try the many demos that come in the Matplotlib source distribution, download the *.tar.gz file and look in the examples subdirectory.

To run the test suite:

- extractthelib/matplotlib/testsorlib/mpl_toolkits/testsdirectoriesfromthesource distribution;
- install test dependencies: pytest, Pillow, MiKTeX, GhostScript, ffmpeg, avconv, ImageMagick, andInkscape;
- run python-mpytest.

THIRD-PARTY DISTRIBUTIONS OF MATPLOTLIB

➢ Scientific Python Distributions

Anaconda and Canopy and ActiveState are excellent choices that "just work" out of the box for Windows, macOS and common Linux platforms. Win Python is an option for Windows users. All of these distributions include Matplotlib and *lots* of other useful (data) science tools.

➢ Linux: using your package manager

If you are on Linux, you might prefer to use your package manager. Matplotlib is packaged for almost every major Linux distribution.

- Debian / Ubuntu: sudoapt-get installpython3-matplotlib
- Fedora: sudodnf installpython3-matplotlib
- Red Hat: sudo yum installpython3-matplotlib
- Arch: sudopacman -Spython-matplotlibv

➢ Installing from source

If you are interested in contributing to Matplotlib development, running the latest source code, or just like to build everything yourself, it is not difficult to build Matplotlib from source. Grab the latest *tar.gz* release file from the PyPI files page, or if you want to develop Matplotlib or just needthe latestbugfixed version, grab the latest git version Install fromsource.

The standard environment variables CC, CXX, PKG_CONFIG are respected. This means you can set them if your toolchain is prefixed. This may be used for cross

compiling.

```
export CC=x86_64-
pc-linux-gnu-gcc
export CXX=x86_64-
pc-linux-gnu-g++
export PKG_CONFIG=x86_64-pc-linux-gnu-pkg-config
```

Once you have satisfied the requirements detailed below (mainly Python, NumPy, libpng and FreeType), you can build Matplotlib.

```
cd matplotlib
python -mpipinstall .
```

We provide a setup.cfg file which you can use to customize the build process. For example, which default backend to use, whether some of the optional libraries that Matplotlib ships with are installed, and so on. This file will be particularly useful to those packaging Matplotlib.

If you have installed prerequisites to nonstandard places and need to inform Matplotlib where they are, edit setupext.py and add the base dirsto the basedir dictionary entry for your sys.platform; e.g., if the header of some required library is in/ some/ path/ include/ someheader.h, put/ some/ path in the base dir list for you rplatform.

➢ Dependencies

Matplotlib requires the following dependencies:

- Python (>=3.5)
- FreeType (>=2.3)

- libpng (>=1.2)
- NumPy (>=1.10.0)
- setuptools
- cycler (>=0.10.0)
- dateutil (>=2.1)
- kiwisolver (>=1.0.0)
- pyparsing

Optionally, you can also install a number of packages to enable better user interface toolkits. See What is a backend? for more details on the optional Matplotlib backends and the capabilities they provide.

- tk (>= 8.3, != 8.6.0 or 8.6.1): for the Tk-basedbackends;
- PyQt4 (>= 4.6) or PySide (>= 1.0.3): for the Qt4-basedbackends;
- PyQt5: for the Qt5-basedbackends;
- PyGObject or pgi (>= 0.0.11.2): for the GTK3-basedbackends;
- wxpython (>= 4): for the WX-basedbackends;
- cairocffi (>= 0.8) or pycairo: for the cairo-basedbackends;
- Tornado: for the WebAggbackend;

For better support of animation output format and image file formats, LaTeX, etc., you can install the following:

- ffmpeg/avconv: for savingmovies;
- ImageMagick: for saving animatedgifs;
- Pillow (>= 3.4): for a larger selection of image file formats: JPEG, BMP, and TIFF imagefiles;
- LaTeX and GhostScript (>=9.0) : for rendering text withLaTeX.

➢ Parts of a Figure

The whole figure. The figure keeps track of all the child Axes, a smattering of 'special' artists (titles, figure legends, etc), and the canvas. (Don't worry too much about the canvas, it is crucial as it is the object that actually does the drawing to get you your plot, but as the user it is more-or-less invisible to you). A figure can have any number of Axes, but to be useful should have atleast one.

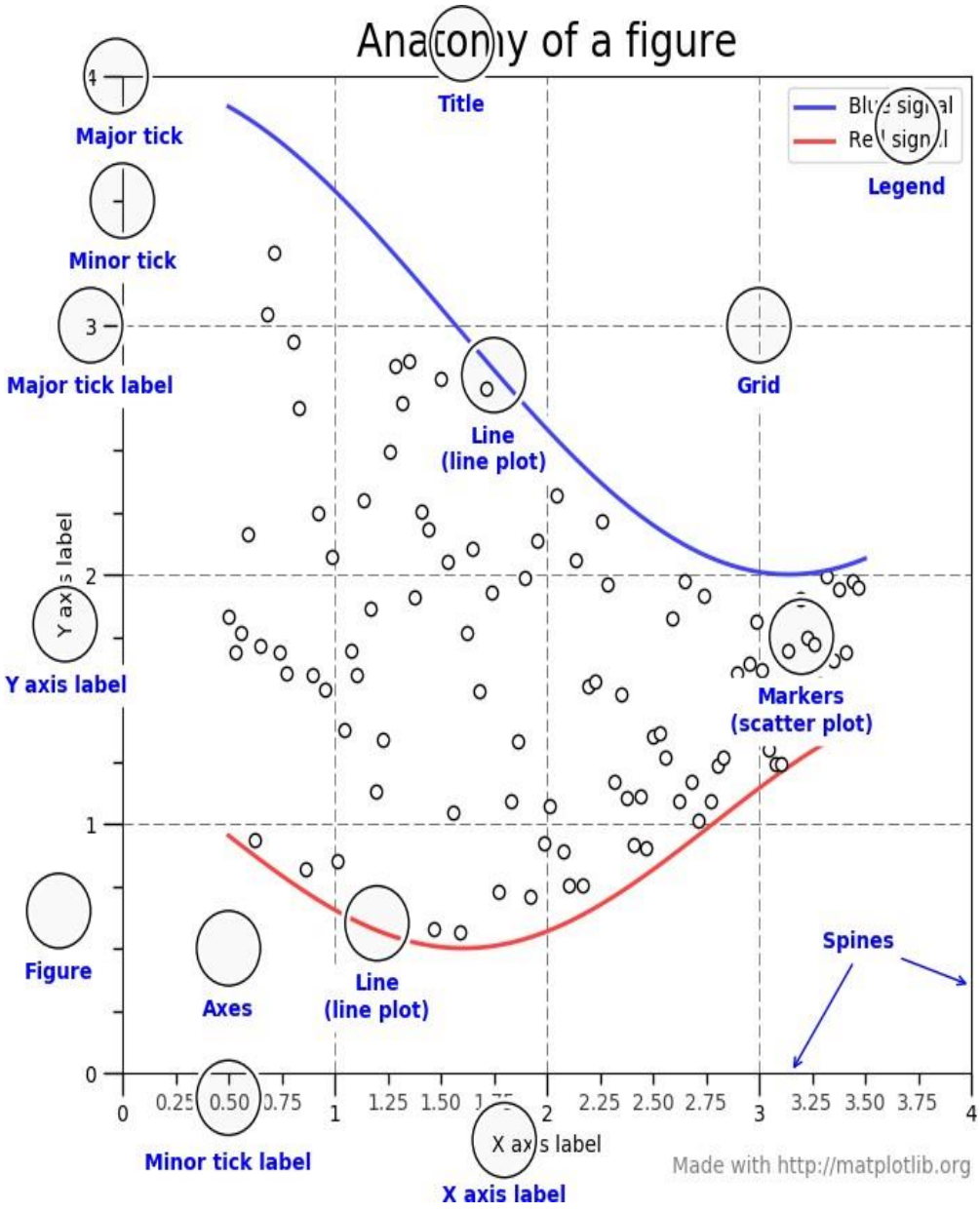


Figure 3.18 -Track of all the child Axes,

3.7.6 KERAS

Keras is a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Keras contains numerous implementations of commonly used neural network building blocks suchas layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

- Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine.
- It also allows use of distributed training of deep learning models on clusters of Graphics Processing Units (GPU).

Guiding principles

- **Modularity:** A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.
- **Minimalism**: Each module should be kept short and simple. Every piece of code should be transparent upon first reading. No black magic: it hurts iteration speed and ability to innovate.
- **Easy extensibility:** New modules are dead simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.
- **Work with Python:** No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

Installation

Keras uses the following dependencies:

- numpy, scipy
- pyyaml
- HDF5 and h5py (optional, required if you use model saving/loading functions)
- Optional but recommended if you use CNNs: cuDNN.

To install Keras, `cd` to the Keras folder and run the install command:

```
sudo python setup.py install
```

You can also install Keras from PyPI:

```
sudo pip install keras
```

➢ **Video Question Answering Model**

Now that we have trained our image QA model, we can quickly turn it into a video QA model. With appropriate training, you will be able to show it a short video (e.g. 100-frame human action) and ask a natural language question about the video (e.g. "what sport is the boy playing?" -> "football").

```
from keras.layers import TimeDistributed
```

```
video_input = Input(shape=(100, 3, 224, 224))
```

```
# this is our video encoded via the previously trained vision_model (weights are
reused)
```

```
encoded_frame_sequence = TimeDistributed(vision_model)(video_input)  # the
output will be a sequence of vectors
encoded_video = LSTM(256)(encoded_frame_sequence)  # the output will be a
vector

# this is a model-level representation of the question encoder, reusing the same
weights as before:
question_encoder = Model(input=question_input, output=encoded_question)

# let's use it to encode the question:
video_question_input = Input(shape=(100,), dtype='int32')
encoded_video_question = question_encoder(video_question_input)

# and this is our video question answering model:
merged = merge([encoded_video, encoded_video_question], mode='concat')
output = Dense(1000, activation='softmax')(merged)
video_qa_model = Model(input=[video_input, video_question_input],
output=output)
```

➢ Models in Keras

There are two types of models available in Keras:

- The Sequential model
- The Model class used with functional API.

These models have a number of methods in common:

- `model.summary()` : prints a summary representation of your model.
- `model.get_config()` : returns a dictionary containing the configuration of the model. The model can be reinstantiated from its config via:

```
config = model.get_config()
```

```
model = Model.from_config(config)
```

```
# or, for Sequential:
```

```
model = Sequential.from_config(config)
```

- model.get_weights(): returns a list of all weight tensors in the model, as Numpy arrays.
- model.set_weights(weights): sets the values of the weights of the model, from a list of Numpy arrays. The arrays in the list should have the same shape as those returned by get_weights().
- model.to_json(): returns a representation of the model as a JSON string. Note that the representation does not include the weights, only the architecture. You can reinstantiate the same model (with reinitialized weights) from the JSON string via:

```
from models import model_from_json

json_string = model.to_json()
model = model_from_json(json_string)
```

- model.to_yaml(): returns a representation of the model as a YAML string. Note that the representation does not include the weights, only the architecture. You can reinstantiate the same model (with reinitialized weights) from the YAML string via:

```
from models import model_from_yaml

yaml_string = model.to_yaml()
model = model_from_yaml(yaml_string)
```

- model.save_weights(filepath): saves the weights of the model as a HDF5 file.
- model.load_weights(filepath, by_name=False): loads the weights of the model from a HDF5 file (created by save_weights). By default, the architecture is expected to be unchanged. To load weights into a different architecture (with some layers in common), use by_name=True to load only those layers with the same name.

CHAPTER-4

SYSTEM DESIGN

4.1 UML DIAGRAMS

4.1.1 Flow Chart

A **flowchart** is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by- step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields

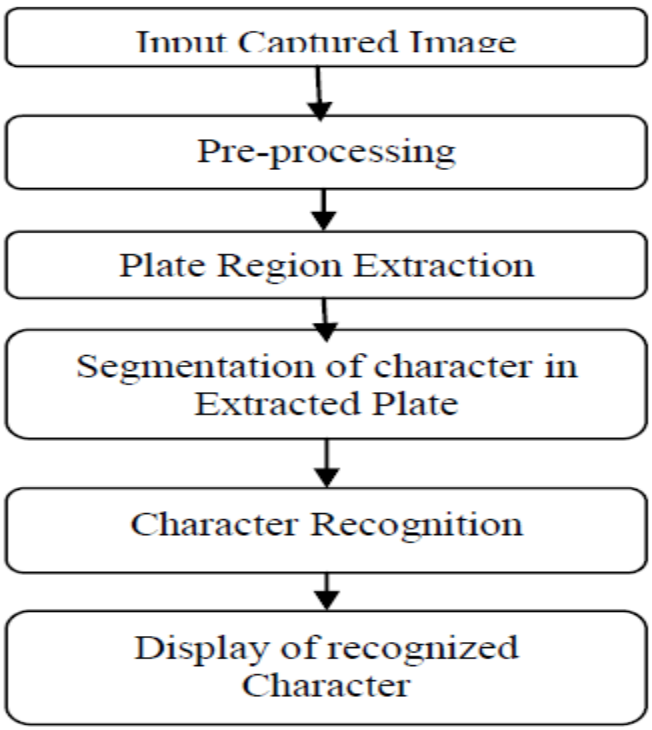
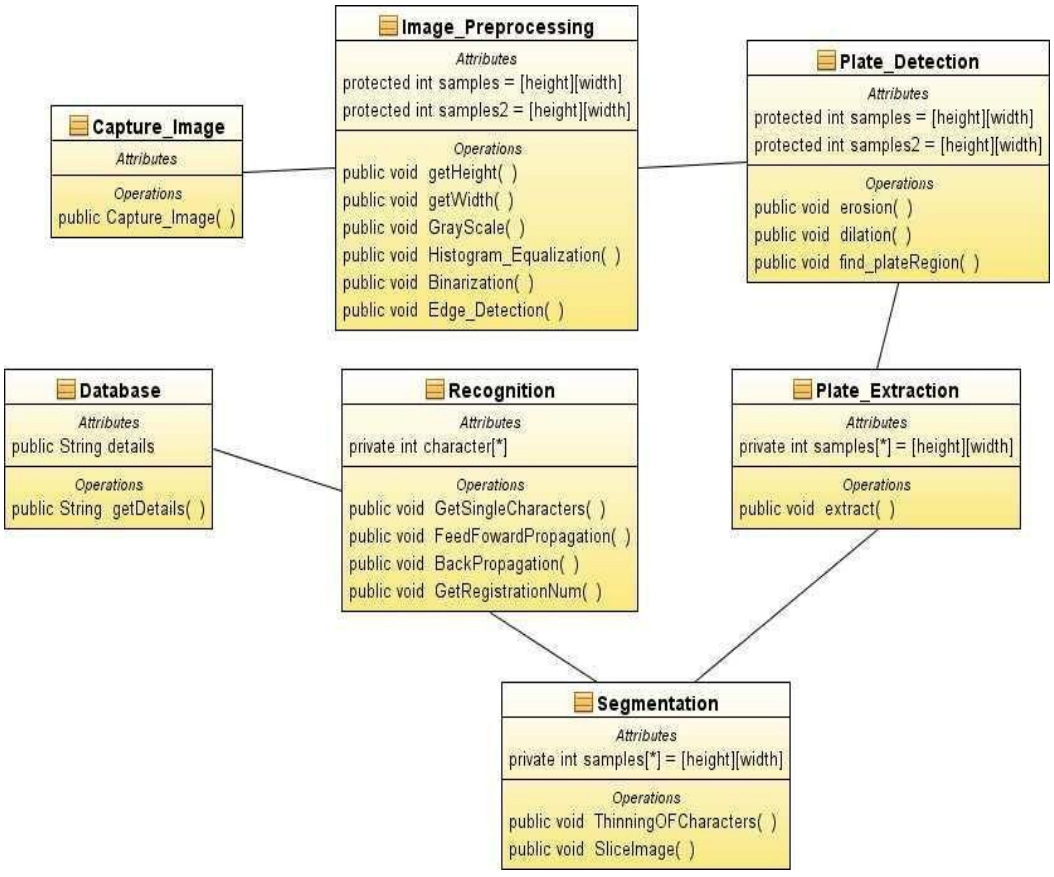


Figure Flow diagram of number plate recognition algorithm

4.1.2 Class Diagram

The class diagram gives a clear picture of all the processes involved in the background in order to carry out the recognition process. It shows all the classes that happens in the background and as well gives a clear relationship on how they relates with one another to help recognize the characters in the plates at the end of the day. The class diagram contains of all the attributes involved in each class or method. It also gives a high clear idea towards the entire processing of the image, how the image is being processes to cater for recognizing the characters.

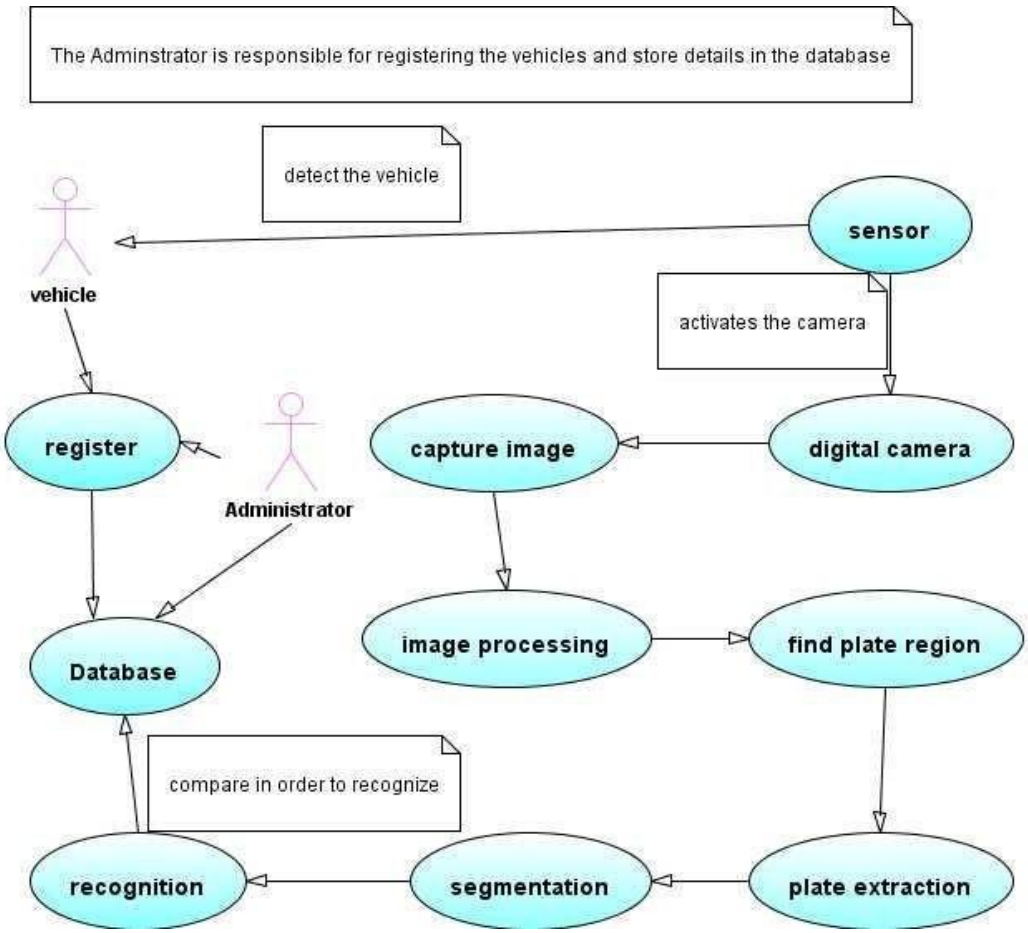


4.1.3 Use Case Diagram

A use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

A use case diagram is usually simple. It does not show the detail of the use cases:

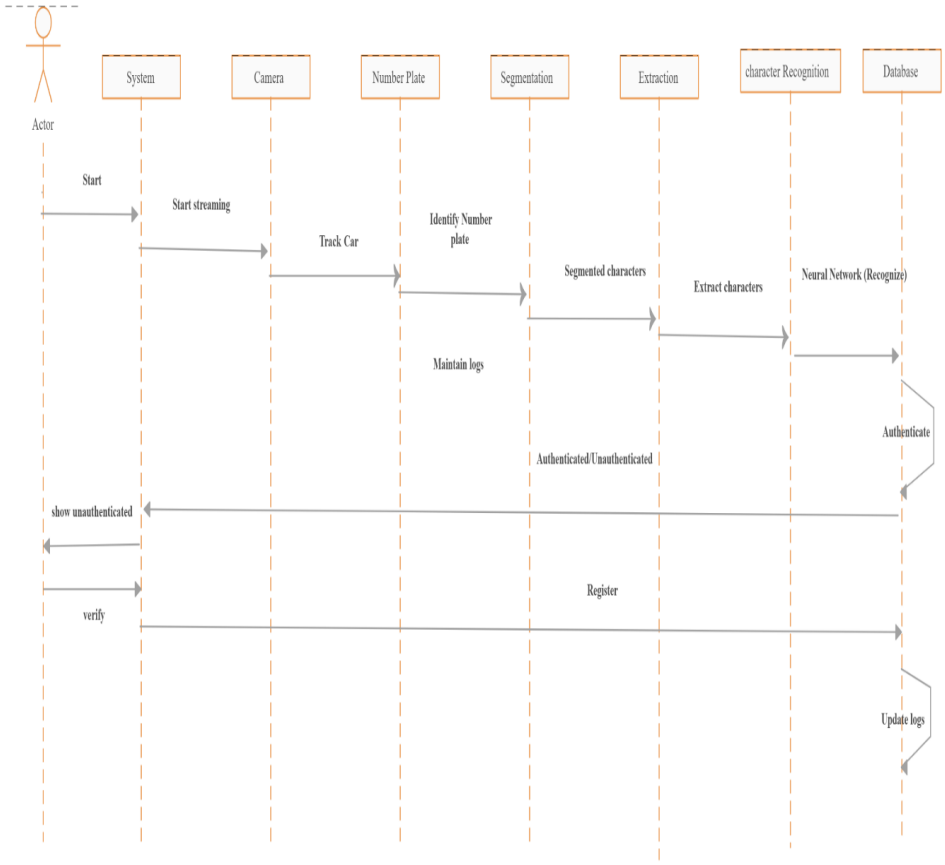
- It only summarizes **some of the relationships** between use cases, actors, and systems.
- It does **not show the order** in which steps are performed to achieve the goals of each use case.



4.1.4 Sequence Diagram

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when. Sequence Diagrams captures:

- The interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams).
- High-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams).

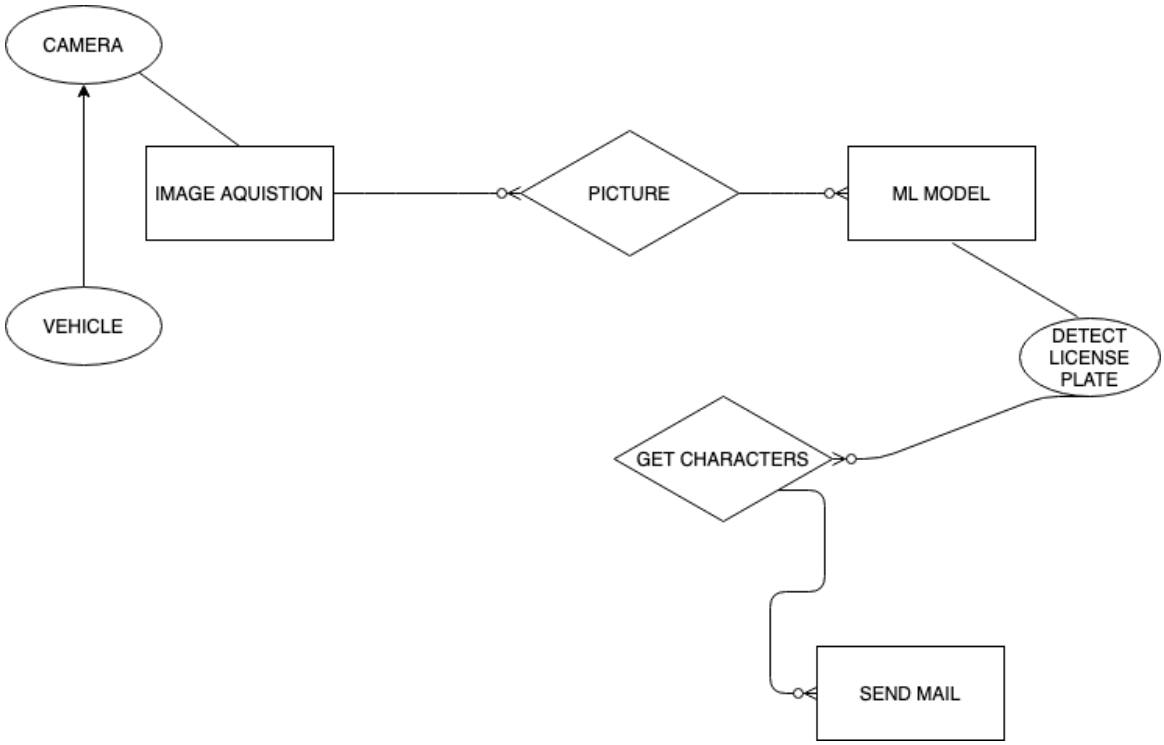


4.1.5 ER Diagram

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

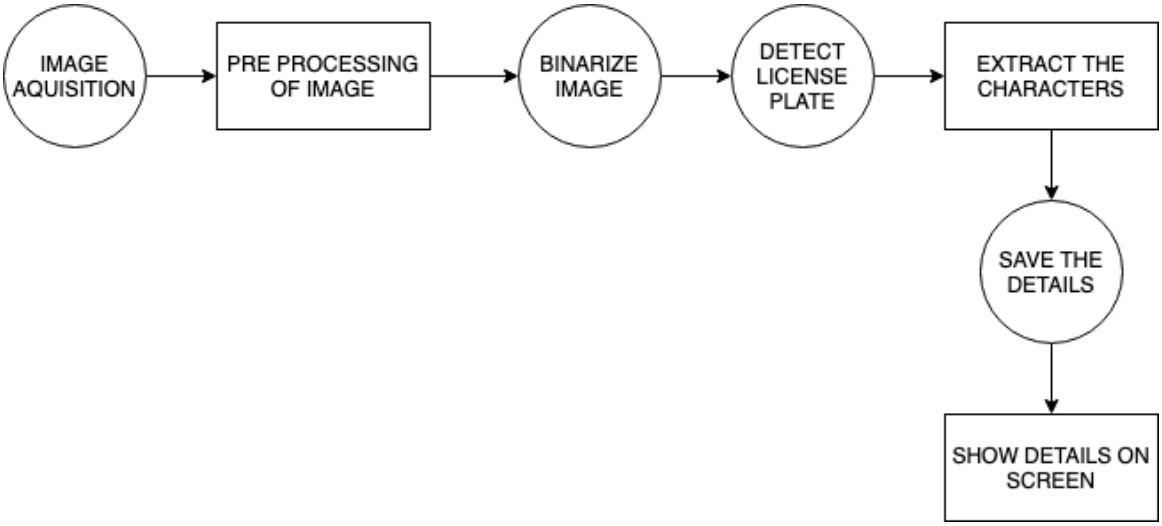
An ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.



4.1.6 Dataflow Diagram

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.



CHAPTER -5

SYSTEM REQUIREMENTS

5.1 SOFTWARE REQUIREMENTS

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed. In this system study the following software requirements used.

- OS:Windws XP + / Linux/ MacOS
- Python 3
- Scikit-learn
- Matplotlib
- Numpy
- Jupyter notebook

5.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. In this system study the following hardware requirements used.

Component	Minimum requirement
Processor	64-bit, i3 processor, 2.5 GHz minimum per core
Ram	8 GB for developer or evaluation use 16 GB for single server and multiple server farm installation for production use
Hard disk	120 GB

Table 5.1-Hardware Requirements

CHAPTER-6

CODING

IMPORTING MODULES

```
import cv2

import numpy as np

from local_utils import detect_lp

from os.path import splitext

from keras.models import model_from_json

from sklearn.preprocessing import LabelEncoder
```

LOADING THE MODEL

```
def load_model(path):
    try:

        path = splitext(path)[0]

        with open('%s.json' % path, 'r') as json_file:

model_json = json_file.read()

        model = model_from_json(model_json, custom_objects={})

model.load_weights('%s.h5' % path)

print("Loading model successfully...")

        return model

    except Exception as e:

        print(e)
```

PASSING THE MODEL

```
wpod_net_path = "wpod-net.json"

wpod_net = load_model(wpod_net_path)
```

LOAD MODEL ARCHITECTURE WEIGHT AND LABELS

```
json_file = open('MobileNets_character_recognition.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)
model.load_weights("License_character_recognition_weight.h5")
print("[INFO] Model loaded successfully...")
labels = LabelEncoder()
labels.classes_ = np.load('license_character_classes.npy')
print("[INFO] Labels loaded successfully...")
```

PERFORMING PREPROCESSING

```
def preprocess_image(img, resize=False):
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img = img / 255
    if resize:
img = cv2.resize(img, (224, 224))
    return img
```

EXTRACTING NUMBER PLATE

```
def get_plate(img, Dmax=608, Dmin=608):
    vehicle = preprocess_image(img)
    ratio = float(max(vehicle.shape[:2])) / min(vehicle.shape[:2])
    side = int(ratio * Dmin)
bound_dim = min(side, Dmax)
```

```
    _, LpImg, _, cor = detect_lp(wpod_net, vehicle, bound_dim, lp_threshold=0.5)
    return vehicle, LpImg, cor
```

CAPTURING VIDEO

```
cap = cv2.VideoCapture("./Plate_examples/Sample2.mp4")
while cap.isOpened():
final_string = ''
    ret, img = cap.read()
    # if frame is read correctly ret is True
    if not ret:
print("Can't receive frame (stream end?). Exiting ...")
    break
    try:
        vehicle, LpImg, cor = get_plate(img)
        if len(LpImg):  # check if there is at least one license image
                    # Scales, calculates absolute values, and converts the result to 8-bit.
plate_image = cv2.convertScaleAbs(LpImg[0], alpha=255.0)
```

CONVERT TO GRAYSCALE AND BLUR THE IMAGE

```
gray = cv2.cvtColor(plate_image, cv2.COLOR_BGR2GRAY)
            blur = cv2.GaussianBlur(gray, (7, 7), 0)
            # Applied inversed thresh_binary
            binary = cv2.threshold(blur, 180, 255,
                                   cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
            kernel3 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
thre_mor = cv2.morphologyEx(binary, cv2.MORPH_DILATE, kernel3)
```

SETTING THE CONTOURS

```
def sort_contours(cnts, reverse=False):
i = 0
boundingBoxes = [cv2.boundingRect(c) for c in cnts]
    (cnts, boundingBoxes) = zip(*sorted(zip(cnts, boundingBoxes),
    key=lambda b: b[1][i], reverse=reverse))
    return cnts
```

FINDING THE CONTOURS

```
    cont, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

  # creat a copy version "test_roi" of plat_image to draw bounding box
test_roi = plate_image.copy()
```

INITIALIZE A LIST WHICH WILL BE USED TO APPEND CHARACTER IMAGE

```
crop_characters = []
  # define standard width and height of character
digit_w, digit_h = 30, 60
  for c in sort_contours(cont):
    (x, y, w, h) = cv2.boundingRect(c)
    ratio = h / w
    if 1 <= ratio <= 3.5:  # Only select contour with defined ratio
      if h / plate_image.shape[0] >= 0.5:
          # Select contour which has the height larger than 50% of the plate
```

DRAW BOUNDING BOX ARROUND DIGIT NUMBER

```
                cv2.rectangle(test_roi, (x, y), (x + w, y + h), (0, 255, 0), 2)

                # Sperate number and gibe prediction

curr_num = thre_mor[y:y + h, x:x + w]

curr_num = cv2.resize(curr_num, dsize=(digit_w, digit_h))

                _, curr_num = cv2.threshold(curr_num, 220, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)

crop_characters.append(curr_num)

        # pre-processing input images and predict with model

        def predict_from_model(image, model, labels):

            image = cv2.resize(image, (80, 80))

            image = np.stack((image,) * 3, axis=-1)

            prediction =
labels.inverse_transform([np.argmax(model.predict(image[np.newaxis, :]))])

            return prediction

        for i, character in enumerate(crop_characters):

            title = np.array2string(predict_from_model(character, model, labels))

final_string += title.strip("'[]")

#printing the output

        print(final_string)

    except:

        pass
```

```
    finally:
        cv2.putText(img, final_string, (100, 150), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 0, 255), 5)
        cv2.imshow("img", img)
        if cv2.waitKey(1) == 27:
            break
cap.release()
cv2.destroyAllWindows()
```

CHAPTER-7

TESTING

7.1 TEST CASES

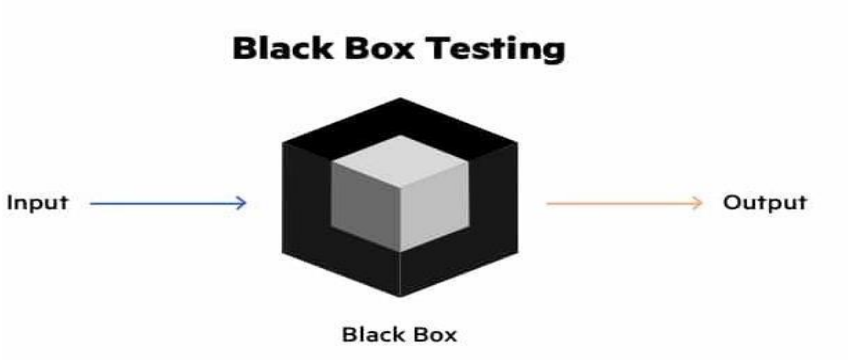
Test cases can be divided in to two types. First one is Positive test cases and second one is negative test cases. In positive test cases are conducted by the developer intention is to get the output. In negative test cases are conducted by the developer intention is to don't get the output.

7.2 BLACK BOX TESTING

Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues.

Black box testing is a powerful testing technique because it exercises a system end-to-end. Just like end-users "don't care" how a system is coded or architected, and expect to receive an appropriate response to their requests, a tester can simulate user activity and see if the system delivers on its promises. Along the way, a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.

An example of a security technology that performs black box testing is Dynamic Application Security Testing (DAST), which tests products in staging or production and provides feedback on compliance and security issues.



Black Box Testing Pros and Cons

No	Pros	Cons
1	Testers do not require technical knowledge, programming or IT skills	Difficult to automate
2	Testers do not need to learn implementation details of the system	Requires prioritization, typically infeasible to test all user paths
3	Tests can be executed by crowdsourced or outsourced testers	Difficult to calculate test coverage
4	Low chance of false positives	If a test fails, it can be difficult to understand the root cause of the issue
5	Tests have lower complexity, since they simply model common user behavior	Tests may be conducted at low scale or on a non-production-like environment

Table 3.1-Pros and Cons of black box testing

7.2.1 Types of Black Box Testing

Black box testing can be applied to three main types of tests: functional, non-functional, and regression testing.

i) Functional Testing

Black box testing can test specific functions or features of the software under test. For example, checking that it is possible to log in using correct user credentials, and not possible to log in using wrong credentials.

Functional testing can focus on the most critical aspects of the software (smoke testing/sanity testing), on integration between key components (integration testing), or on the system as a whole (system testing).

ii) Non-Functional Testing

Black box testing can check additional aspects of the software, beyond features and functionality. A non-functional test does not check “if” the software can perform a specific action but “how” it performs that action.

Black box tests can uncover if software is:

- Usable and easy to understand for its users
- Performance under expected or peak loads
- Compatible with relevant devices, screen sizes, browsers or operating systems
- Exposed to security vulnerabilities or common security threats

7.2.2 Techniques of Black Box Testing

1. **Equivalence Partitioning:**

   Testers can divide possible inputs into groups or "partitions", and test only one example input from each group. For example, if a system requires a user's birth date and provides the same response for all users under the age of 18, and a different response for users over 18, it is sufficient for testers to check one birth date in the "under 18" group and one date in the "over 18" group.

2. **Boundary Value Analysis:**

   Testers can identify that a system has a special response around a specific boundary value. For example, a specific field may accept only values between 0 and 99. Testers can focus on the boundary values (-1, 0, 99 and 100), to see if the system is accepting and rejecting inputs correctly.

3. **Decision Table Testing:**

   Many systems provide outputs based on a set of conditions. Testers can then identify "rules" which are a combination of conditions, identify the outcome of each rule, and design a test case for each rule.

For example, a health insurance company may provide different premium based on the age of the insured person (under 40 or over 40) and whether they are a smoker or not. This generates a decision table with four rules and up to four outcomes—below is an example with three possible outcomes.

7.3 WHITE BOX TESTING

White-box testing's basic procedures require the tester to have an in-depth knowledge of the source code being tested. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analyzed for test cases to be created. The following are the three basic steps that white-box testing takes in order to create test cases:

1. Input involves different types of requirements, functional specifications, detailed designing of documents, proper source code and security specifications This is the preparation stage of white-box testing to lay out all of the basic information.
2. Processing involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.
3. Output involves preparing final report that encompasses all of the above preparations and result

7.3.1 Types of White Box Testing

White box testing encompasses several testing types used to evaluate the usability of an application, block of code or specific software package. There are listed below

I. **Unit Testing:**

It is often the first type of testing done on an application. Unit Testing is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object and test it to make sure it works before continuing Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.

II. **Testing for Memory Leaks**:

Memory leaks are leading causes of slower running applications. A

QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.

7.3.2 Techniques of White Box Testing

1. **Data Flow Technique:**

   Data flow testing is used to analyze the flow of data in the program. It is the process of collecting information about how the variables flow the data in the program. It tries to obtain particular information of each particular point in the process.

2. **Control Flow Testing:**

   Control flow testing determines the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. Test cases represented by the control graph of the program.

3. **Branch Coverage Technique:**

   Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once.

CHAPTER-8

OUTPUT SCREENS


Output ANPR
KA297999
KA 29 Z 999


MH 20 EE 7601
(x=389, y=3) – R:36 G:58 B:97

CHAPTER-9

CONCLUSION

An efficient and convenient method for recognizing license plate based on Convolutional neural network have been proposed. The proposed system is useful in various scenarios such as parking system, vehicle theft control, vehicle speed monitoring, reducing traffic congestion, automatic toll tax collection etc. The system will capture the images of vehicles and extracts the number plate from the image automatically and then character segmentation and recognition will be performed in order to recognize the characters written on the number plate. The CNN based number plate recognition system is found to be more efficient and accurate than all the other existing ANPR models.

Currently, ANPR works on Template Matching Technique which is prone to the following challenges: circumvention, plate inconsistency and jurisdictional differences, Accuracy and measurement of ANPR system performance, Speed of the vehicles, incorrect cameras, weather conditions, poor on site traffic management resulting in missed plates/vehicles, excessive skew angles causing recognition issues. These challenges faced by the existing system clearly demands for a better one.

The system works satisfactorily for wide variations in illumination conditions and different types of number plates commonly found in India. It is definitely a better alternative to the existing proprietary systems, even though there are known restrictions.

As compared to previous approaches, license plates have been recognized from full camera still as well as a parking video with noise. On the dataset of 4800 car images, the accuracy obtained is 91% on number plate extraction from images, 93% on character recognition. Proposed ALPR system has also been applied to vehicle videos shot at parking exits. Overall 85% accuracy was obtained in real-time license number recognition from these videos.

CHAPTER-10

FUTURE ENCHANCEMENT

Although we can see that so many algorithms have been implemented in various previous projects, in order to make a robust system for automatic license plate recognition, there are still many loop holes left in the system which can be filled in order to make the system more future-proof and reliable.

In future we can include face acknowledgment and connect the entire framework with criminal database and recognize passing criminal out laws.

Our project however works on the simple font styles which is being used normally on license plates of the cars as per the rules made by the governing bodies of traffic department. But in order to handle the cases where people don't follow these rules, it can be handled in future projects being implemented in this field of license plate recognition.

Some of the various fields which can be explored in this project are as follows:

- Car model recognition
- Multi-lingual character recognition
- Fancy character recognition etc.

CHAPTER-11

REFERENCES

1. Ms. Shilpi Chauhan and Vishal Srivastava “Matlab Based Vehicle Number Plate Recognition”- Research Scholar and Associate Professor, Dept. of CSE, Arya Institute of Engineering and IT, Jaipur, India.

2. Subhadhira,S., Juithonglang, U., Sakulkoo, P., &Horata, P. (2014). “License plate recognition application using extreme learning machines”, Third ICT International Student Project Conference (ICTISPC).2014

3. Michael Nielsen. Neural Networks and Deep Learning. Online,accessed.26-April2018.
   http://neuralnetworksanddeeplearning.com/chap1.html.

4. Niklas Donges. Pros and Cons of Neural Networks. Online, accessed 26- July-2018.
   https://towardsdatascience.com/hype-disadvantages-of-neural-networks6af04904ba5b.

5. Anand Sumatilal Jain, Jayshree M. Kundargi; Automatic License plate Recognition Using Artificial Neural Network IRJET Volume: 02 Issue: 04 | July-2015;
   e-ISSN: 2395-0056; p-ISSN: 2395-0072

6. Amarjot Singh, Ketan Bacchuwar, and Akshay Bhasin; A Survey of OCR Applications; International Journal of Machine Learning and Computing, Vol. 2, No. 3; June2012

7. Indian Central Motor Vehicle Act, Bear Act, Rule no.49,50.

8. Zhao, Z.; Yang, S.; Ma, X. Chinese License Plate Recognition Using a Convolutional Neural Network. In Proceedings of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, Wuhan, China, 19–20 December 2008; IEEE: New York, NY, USA, 2008; pp. 27–30. [CrossRef].

9. Alam, N.-A.; Ahsan, M.; Based, M.A.; Haider, J. Intelligent System for Vehicles Number Plate Detection and Recognition Using Convolutional Neural Networks.Technologies2021, 9, 9.
https://doi.org/10.3390/technologies9010009

10. Shridhar, M.; Miller, J. W V; Houle, G.; Bijnagte, L., "Recognition of license plate images: issues and perspectives," Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on, vol., no., pp.17,20, 20 -22 Sep1999.

11. S.V. Rice, F.R. Jenkins, T.A. Nartker, The Fourth Annual Test of OCR Accuracy, Technical Report 95-03. Information Science Research Institute, UniversityofNevada,LasVegas,(1995).
http://opencv.willowgarage.com/documentation/python/

12. A.Conci, J. E. R. de Carvalho, T. W. Rauber, A Complete System for Vehicle Plate Localization, Segmentation and Recognition in Real Life Scene, IEEE LATIN AMERICA TRANSACTIONS, VOL. 7, NO.5,SEPTEMBER 2009.

13. S.V. Rice, F.R. Jenkins, T.A. Nartker, The Fourth Annual Test of OCR Accuracy, Technical Report 95-03. Information Science Research Institute, University of Nevada, LasVegas,(1995).
http://opencv.willowgarage.com/documentation/python/

14. A.Conci, J. E. R. de Carvalho, T. W. Rauber, A Complete System for Vehicle Plate Localization, Segmentation and Recognition in Real Life Scene, IEEE

LATIN AMERICA TRANSACTIONS, VOL. 7, NO.5,SEPTEMBER 2009

15. Prathamesh Kulkarni, Ashish Khatri, Prateek Banga, Kushal Shah, AutomaticNumberPlateRecognition(ANPR)SystemforIndianconditions

16. S.V. Rice, F.R. Jenkins, T.A. Nartker, The Fourth Annual Test of OCR Accuracy, Technical Report 95-03. Information Science Research Institute, University of Nevada, LasVegas,(1995).
http://opencv.willowgarage.com/documentation/python/

17. A.Conci, J. E. R. de Carvalho, T. W. Rauber, A Complete System for Vehicle Plate Localization, Segmentation and Recognition in Real Life Scene, IEEE LATIN AMERICA TRANSACTIONS, VOL. 7, NO.5,SEPTEMBER 2009
http://timesofindia.indiatimes.com/city/delhi/40-vehicles-stolen-in-Delhi-every-day/articleshow/14619149.cms

18. Amarjot Singh, Ketan Bacchuwar, and Akshay Bhasin; A Survey of OCR Applications; International Journal of Machine Learning and Computing, Vol. 2, No. 3; June2012

19. S.N.Sivanandam, S.N.Deepa"Principles of Soft Computing" Second Edition, WileyPublication.Indian Central Motor Vehicle Act, Bear Act, Rule no.49,50.