**TRANSFER LEARNING BASED LICENSE PLATE DETECTION AND RECOGNITION SYSTEM**

**ABSTRACT**

Recognizing a vehicle’s plate number is equivalent to identifying its identity and associating it to any good and bad road behaviors, e.g., driving at the prescribed road speed. Such recognition and identification is an integral part of an intelligent transportation system and future smart city. In this paper, we propose an integrated system of identifying a vehicle’s plate number. Experimental results illustrate a relatively higher percentage of plate number identification and recognition. This project describes the license plate number recognition using deep learning algorithm. Optical character recognition is implemented for recognition of letter and character.

**CHAPTER 1**

**INTRODUCTION**

In recent years, with the continuous improvement of China's social and economic level, the popularization rate of China's automobile has greatly increased, and the status of China's automobile industry in the world is also rising. However, due to the rapid development of the modern transportation industry and urban construction industry, there is a significant increase in the number of motor vehicles, more and more traffic safety accidents have occurred in China, especially with the development of expressway, the injury and fatality rate of traffic accidents is greatly increased. At present, the main solutions to traffic problems are as follows: controlling the traffic demand, such as taking measures to reduce the number of motor vehicles, but this method is not conducive to long-term development; building more transportation infrastructure, but this way is limited by the financial shortage, unreasonable road design and other factors; adopting intelligent transportation system, which is a ground transportation system based on computer technology, artificial intelligence technology, and information technology. Fig 1.1 illustrates number plate



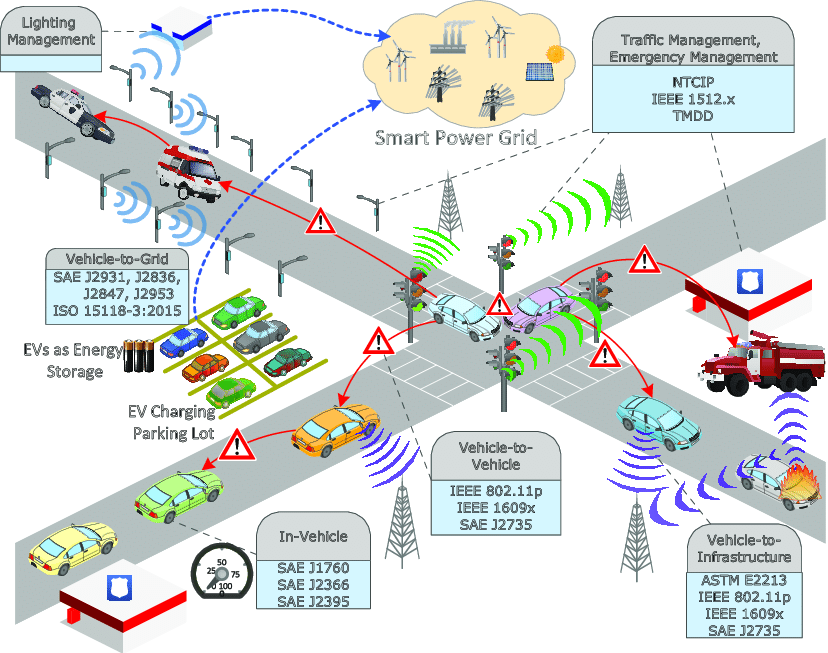
The license plate recognition system, which uses digital image processing, computer graphics, computer vision, character recognition, and other technologies, is an important exploration in modern intelligent transportation. For example, the recognition system can be utilized for managing park facilities, monitoring unauthorized vehicles entering private areas, detecting stolen vehicles, controlling traffic volume, ticketing speeding vehicles, and so on. In license plate recognition technology, detection is the key.

Foreign research and development of license plate recognition system are early, and the recognition technology has been very mature, and it has been widely used, their years of experience has also developed a variety of license plate recognition relating to products. Because the products developed by western companies are basically aimed at local license plate recognition, and most of them can only recognize foreign language characters, China cannot fully adopt foreign license plate recognition technology but can learn from its advantages.

Compared with foreign countries, the research on license plate recognition technology in China is relatively late, Chinese companies developing license plate recognition systems include Chengdu Zhenzhi Technology Development Co., Ltd., which has developed Huoyan Zhenjing license plate recognition system; Beijing Wentong Technology Co., Ltd., which has developed Wentong automobile license plate recognition system; Beijing Zhitong Video Technology Development Co., Ltd., which has developed license plate recognition integrated machine, etc. Many Chinese researchers have carried out in-depth research in the field of license plate recognition, developed a wealth of related algorithms, and achieved useful progress. However, in China, the background color of the license plate and the color of the character bar frame are different. In addition, due to the interference of complex environments and the inability to obtain clear pictures, there is no general open algorithm for license plate recognition in arbitrary environments in China.

* 1. **Intelligent transportation system**

An intelligent transportation system (ITS) is an advanced application that aims to provide innovative services relating to different modes of transport and traffic management and enable users to be better informed and make safer, more coordinated, and 'smarter' use of transport networks. Fig: 1.1. Intelligent transportation system



Some of these technologies include calling for emergency services when an accident occurs, using cameras to enforce traffic laws or signs that mark speed limit changes depending on conditions.

* 1. **License plate detection**

Automatic number-plate recognition (ANPR; see also other names below) is a technology that uses optical character recognition on images to read vehicle registration plates to create vehicle location data. It can use existing closed-circuit television, road-rule enforcement cameras, or cameras specifically designed for the task. ANPR is used by police forces around the world for law enforcement purposes, including to check if a vehicle is registered or licensed. It is also used for electronic toll collection on pay-per-use roads and as a method of cataloging the movements of traffic, for example by highways agencies. Fig: 1.2 License Plate



**INTRODUCTION-IMAGE PROCESSING**

**GENERAL**

The term digital image refers to processing of a two dimensional picture by a digital computer. In a broader context, it implies digital processing of any two dimensional data. A digital image is an array of real or complex numbers represented by a finite number of bits. An image given in the form of a transparency, slide, photograph or an X-ray is first digitized and stored as a matrix of binary digits in computer memory. This digitized image can then be processed and/or displayed on a high-resolution television monitor. For display, the image is stored in a rapid-access buffer memory, which refreshes the monitor at a rate of 25 frames per second to produce a visually continuous display.

**DIGITIZER**

**MASS STORAGE**

**HARD COPY DEVICE**

**DISPLAY**

**IMAGE PROCESSOR**

**DIGITAL COMPUTER**

**OPERATOR CONSOLE**

**FIG 1.1 BLOCK DIAGRAM FOR IMAGE PROCESSING SYSTEM**

* **DIGITIZER**

A digitizer converts an image into a numerical representation suitable for input into a digital computer. Some common digitizers are

* Microdensitometer
* Flying spot scanner
* Image dissector
* Videocon camera
* Photosensitive solid- state arrays.
* **IMAGE PROCESSOR**

An image processor does the functions of image acquisition, storage, preprocessing, segmentation, representation, recognition and interpretation and finally displays or records the resulting image. The following block diagram gives the fundamental sequence involved in an image processing system.

**PROBLEM DOMAIN**

**KNOWLEDGE**

**BASE**

**SEGMENTATION**

**PREPROCESSING**

**IMAGE ACQUISITION**

**RECOGNITION & INTERPRETATION**

**REPRESENTATION & DESCRIPTION**

**RESULT**

**FIG 1.2 BLOCK DIAGRAM OF FUNDAMENTAL SEQUENCE INVOLVED IN AN IMAGE PROCESSING SYSTEM**

As detailed in the diagram, the first step in the process is image acquisition by an imaging sensor in conjunction with a digitizer to digitize the image. The next step is the preprocessing step where the image is improved being fed as an input to the other processes. Preprocessing typically deals with enhancing, removing noise, isolating regions, etc. Segmentation partitions an image into its constituent parts or objects. The output of segmentation is usually raw pixel data, which consists of either the boundary of the region or the pixels in the region themselves. Representation is the process of transforming the raw pixel data into a form useful for subsequent processing by the computer. Description deals with extracting features that are basic in differentiating one class of objects from another. Recognition assigns a label to an object based on the information provided by its descriptors. Interpretation involves assigning meaning to an ensemble of recognized objects. The knowledge about a problem domain is incorporated into the knowledge base. The knowledge base guides the operation of each processing module and also controls the interaction between the modules. Not all modules need be necessarily present for a specific function. The composition of the image processing system depends on its application. The frame rate of the image processor is normally around 25 frames per second.

* **DIGITAL COMPUTER**

Mathematical processing of the digitized image such as convolution, averaging, addition, subtraction, etc. are done by the computer.

* **MASS STORAGE**

The secondary storage devices normally used are floppy disks, CD ROMs etc.

* **HARD COPY DEVICE**

The hard copy device is used to produce a permanent copy of the image and for the storage of the software involved.

* **OPERATOR CONSOLE**

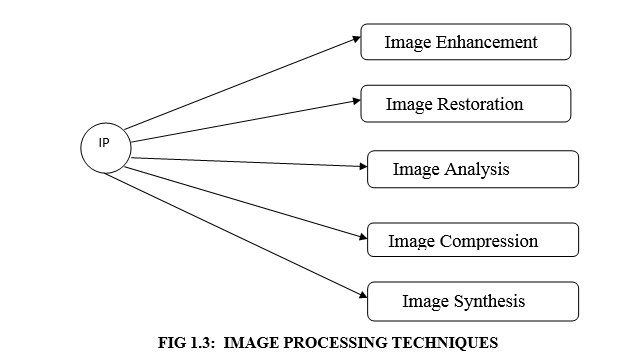
The operator console consists of equipment and arrangements for verification of intermediate results and for alterations in the software as and when require. The operator is also capable of checking for any resulting errors and for the entry of requisite data.

**IMAGE PROCESSING FUNDAMENTAL**

Digital image processing refers processing of the image in digital form. Modern cameras may directly take the image in digital form but generally images are originated in optical form. They are captured by video cameras and digitalized. The digitalization process includes sampling, quantization. Then these images are processed by the five fundamental processes, at least any one of them, not necessarily all of them.

**IMAGE PROCESSING TECHNIQUES**

This section gives various image processing techniques.



* **IMAGE ENHANCEMENT**

Image enhancement operations improve the qualities of an image like improving the image’s contrast and brightness characteristics, reducing its noise content, or sharpen the details. This just enhances the image and reveals the same information in more understandable image. It does not add any information to it.

* **IMAGE RESTORATION**

Image restoration like enhancement improves the qualities of image but all the operations are mainly based on known, measured, or degradations of the original image. Image restorations are used to restore images with problems such as geometric distortion, improper focus, repetitive noise, and camera motion. It is used to correct images for known degradations.

* **IMAGE ANALYSIS**

Image analysis operations produce numerical or graphical information based on characteristics of the original image. They break into objects and then classify them. They depend on the image statistics. Common operations are extraction and description of scene and image features, automated measurements, and object classification. Image analyze are mainly used in machine vision applications.

* **IMAGE COMPRESSION**

Image compression and decompression reduce the data content necessary to describe the image. Most of the images contain lot of redundant information, compression removes all the redundancies. Because of the compression the size is reduced, so efficiently stored or transported. The compressed image is decompressed when displayed. Lossless compression preserves the exact data in the original image, but Lossy compression does not represent the original image but provide excellent compression.

* **IMAGE SYNTHESIS**

Image synthesis operations create images from other images or non-image data. Image synthesis operations generally create images that are either physically impossible or impractical to acquire.

**1.1.3 Image types**

There are several ways of encoding the information in an image.

1. Binary image
2. Grayscale image
3. Indexed image
4. True color or RGB image

* **Binary image**

Each pixel is just blackor white. Since there are only two possible values for each pixel (0, 1), we only need one bitper pixel.

* **Grayscale image**

Each pixel is a shade of gray, normally from 0 (black) to 255(white). This range means that each pixel can be represented by eight bits, or exactly one byte. Other grayscale ranges are used, but generally they are a power of 2.

* **Indexed image**

An indexed image consists of an array and a color map matrix. The pixel values in the array are direct indices into a color map. By convention, this documentation uses the variable name X to refer to the array and map to refer to the color map.

* **True Color or RGB image**

Each pixel has a particular color; that color is described by the amount of red, green and blue in it. If each of these components has a range 0–255, this gives a total of 2563different possible colors. Such an image is a “stack” of three matrices; representing the red, green and bluevalues for each pixel. This means that for every pixel there correspond 3 values.

**APPLICATIONS of image processing**

Image processing has an enormous range of applications; almost every area of science and technology can make use of image processing methods. Here is a short list just to give some indication of the range of image processing applications.

* **DOCUMENT PROCESSING**

It is used in scanning, and transmission for converting paper documents to a digital image form, compressing the image, and storing it on magnetic tape. It is also used in document reading for automatically detecting and recognizing printed characteristics.

* **Medicine**

Inspection and interpretation of images obtained from X-rays, MRI or CAT scans, analysis of cell images, of chromosome karyotypes. In medical applications, one is concerned with processing of chest X-rays, cineangiograms, projection images of transaxial tomography and other medical images that occur in radiology, nuclear magnetic resonance (NMR) and ultrasonic scanning. These images may be used for patient screening and monitoring or for detection of tumors’ or other disease in patients.

* **Industry**

Automatic inspection of items on a production line, inspection of paper samples.

* **DEFENSE/INTELLIGENCE**

It is used in reconnaissance photo-interpretation for automatic interpretation of earth satellite imagery to look for sensitive targets or military threats and target acquisition and guidance for recognizing and tracking targets in real-time smart-bomb and missile-guidance systems.

* **RADAR IMAGING SYSTEM**

Radar and sonar images are used for detection and recognition of various types of targets or in guidance and maneuvering of aircraft or missile systems.

* **Agriculture**

Satellite/aerial views of land, for example to determine how much land is being used for different purposes, or to investigate the suitability of different regions for different crops, inspection of fruit and vegetables distinguishing good and fresh produce from old.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 TITLE:** Adaptive Cruise Control Employing Taillight Tracking for a Platoon of Autonomous Vehicles

**AUTHOR:** Gian Paolo T. Mayuga1 and Elmer R. Magsino2

**YEAR:** 2019

**DESCRIPTION:** Traffic congestions in urban cities unwantedly form platoons of vehicles running at low speeds. For vehicles operated by human drivers, reaction to speeding up or down requires some time, thus, increasing travel time. In this study, we present an adaptive cruise control for a group of autonomous vehicles that follow each other. We propose a taillight tracking system by utilizing low-cost dashboard cameras for detecting the position of the lead vehicle and then allow autonomous vehicles to correctly accelerate or decelerate depending on the nature of traffic. This is achieved by detecting the leading vehicle’s taillight via linear AND-ing of the the RGB and HSV color model representations. We evaluate the proposed system by employing real captured traffic images and tested by utilizing mobile robots for the platoon of vehicles testing.

**2.2 TITLE:** An Integrated Automatic Number Plate Recognition for Recognizing Multi Language Fonts

**AUTHOR:** Arun Vaishnav1, Manju Mandot2

**YEAR:** 2019

**DESCRIPTION:** Automatic Number Plate Recognition (ANPR) is a system for automatic recognition of license number plates. ANPR system uses Optical Character Recognition (OCR) technique for reading the license number plates. Template matching algorithm is applied for obtaining correct matches. English letters are used for registration mark and Aerobic numbers for figures. Ingeneral the font faces used for number plates include Arial, Times New Roman, Calibri, Cambria and Kruti Dev. The present system may yield erratic results for number plates with fonts that are not stored in the template database (DB). ANPR has an ability to recognize and detect number plates with wide range of patterns. In the present work we address the integration of English and Hindi characters together that are used for number plates. We also mention the results and necessary environmental conditions for optimum results of the present system. The Morphological operation and OCR technique are employed for number plate recognition. The images used for testing and training purpose are taken from open source as well as clicking the photos by a good resolution camera (with resolution 8 Mega Pixel or more).

**2.3 TITLE**: Novel and Real-Time GPU Accelerated Speed Violation Detection and License Plate Identification System

**AUTHOR:** Udaya Dampage, KKN Hasantha, HADS Gimhana

**YEAR:** 2021

**DESCRIPTION:** a real-time solution by fully automating the process of detecting the speed violation and the license plates of the offenders is proposed in this paper. A vehicle approaching a specific area will be automatically identified and tracked from a reference starting point. Within the covered range of the camera according to the traffic density present at that instance, the maximum speed for a vehicle is estimated and the vehicles that exceed the stipulated limit are identified as a violation. The core part of the proposed system is license plate recognition. To properly extract the license plate with the best view to proceed with the identification process is another problem that needs to be focused on. We utilized deep neural networks in a novel way for the aforesaid purpose. As these neural networks consist of numerous parameters, we utilized GPU for processing to gain smoothness in real-time. Using our novel segmentation free license plate identification method which utilizes object detection principle to fully capture the speed violation along with its offender.

**2.4 TITLE:** License Number Plate Recognition using Template Matching and Bounding Box Method License Number Plate Recognition using Template Matching and Bounding Box Method

**AUTHOR:** Ira Kusumadewi, Chri

**YEAR:** 2019

**DESCRIPTION:** Recognition of vehicle number plates is very important in the field of transportation security systems. Template matching is one of the most popular techniques used for number plate recognition. This algorithm works by evaluating the image pattern that will be recognized and then comparing it with the image pattern in the template. A number plate is an object in an image that has many characters which usually consists of numbers and letters. Character recognition cannot be done simultaneously in an image. A number plate object in the image needs to be done early processing, normalization, and segmentation to recognize the character. While to separate each character, a bounding box algorithm is used. This technique is relatively simple and has fast computing. So this study examines a combination of template matching and bounding box methods to detect Indonesian car license plates. Based on the tests that have been done, the accuracy reaches 80%.

**2.5 TITLE:** Real-time Jordanian license plate recognition using deep learning

**AUTHOR:** Salah Alghyaline

**YEAR:** 2020

**DESCRIPTION:** Countries have different specifications for License Plates (LPs), therefore developing one Automatic license plate recognition (ALPR) system that works well for all LPs types is a difficult task. This paper aims to develop an accurate ALPR for Jordanian LPs. Two-stage Convolutional Neural Networks (CNNs) are used in the proposed approach, the CNNs are based on the YOLO3 framework. The sizes of LPs’ characters are very small compared with the frame size, therefore the YOLO3 network architecture is modified to a shallow network to detect small objects. The proposed approach uses temporal information from different frames to remove false predictions. A set of arrays data structure is used to track the vehicles’ LPs and eliminate incorrect ones. To my knowledge, the proposed approach represents the first end-to-end Jordanian ALPR that processes video stream in real-time. The dataset is available online and includes many real videos for moving vehicles in Jordan. Two well-known commercial software packages are used for comparisons. The experimental results in real videos from YouTube show that the proposed approach is very efficient in recognizing the Jordanian license plates and achieved 87% recognition accuracy, whereas the commercial systems have recognition accuracies that are less than 81%.

**2.6 TITLE:** License Plate Recognition System Using Artificial Neural Networks

**AUTHOR:** Ibrahim Turkyılmaz and Kirami Kaçan

**YEAR:** 2019

**DESCRIPTION: (**LPRS) is proposed in this work. The proposed LPRS is composed of the following three main stages: (i) plate region determination, (ii) character segmentation, and (iii) character recognition. During the plate region determination stage, the image is enhanced by image processing algorithms to increase system performance. The rectangular license plate region is obtained using edge-based image processing methods on the binarized image. With the help of skew correction, the plate region is prepared for the character segmentation stage. Characters are separated from each other using vertical projections on the plate region. Segmented characters are prepared for the character recognition stage by a thinning process.

**2.7 TITLE**: License Plate Recognition in Urban Road Based on Vehicle Tracking and Result Integration

**AUTHOR:** S Liping Zhu, Shang Wang, Chengyang Li and Zhongguo Yang

**YEAR:** 2019

**DESCRIPTION:** Multiple surveillance cameras provide huge video resources that need further mining to collect traffic stream data such as license plate recognition (LPR). However, these surveillance cameras have limited spatial resolution, which may not always suffice to precisely recognize license plates by existing LPR systems. This work is focused on the LPR method in low-quality images from surveillance video screenshots on urban road. The methodology we proposed is based on vehicle tracking and result integration, and we recognize the plate with an end-to-end method without character segmentation. First, we track each vehicle to get vehicle tracking sequence. Moreover, a plate detector based on an object detection framework is trained to detect license plates of each vehicle from the sequence and a license plate sequence is formed. In addition, an end-to-end convolutional neural network architecture is applied to recognize license plates from the sequence. Finally, we integrate the recognition result of continuous frames to get the final result. Evaluation results on multiple datasets show that our method significantly outperforms others without segmentation or integration in real traffic scene.

**2.8 TITLE:** Automatic number plate recognition using deep learning

**AUTHOR:** V.Gnanaprakash, N.Kanthimathi, N.Saranya

**YEAR:** 2019

**DESCRIPTION:** The number of vehicles on road has been increased. Tracking of individual vehicle becomes a very challenging task with the massive growth in the vehicular sector every day. This paper suggests an automated vehicle tracking system for the fast moving vehicles with the help of the surveillance cameras on the roadside. The process of getting CCTV footage in the real time background is very tedious process. To cater to this problem, an efficient deep learning model such as You Only Look Once (YOLO) is used for object detection. The proposed work consists of four main steps. In the first step, video footage is converted into images and the car is detected from each of the frames. In the next step, license plate is detected from the detected cars. In the final step, the number plate characters reading are recognized from the detected number plates. The proposed deep learning model uses ImageAI library to make the training process easier. Tamil Nadu license plate images are used to analyse the performance of the model. The accuracy of 97% is achieved for car detection, accuracy of 98% is achieved for number plate localization and accuracy of 90% achieved for character recognition.

**2.9 TITLE:** Roadside Unit Allocation for Fog-based Information Sharing in Vehicular Networks

**AUTHOR:** Elmer R. Magsino, Ivan Wang-Hei Ho

**YEAR:** 2019

**DESCRIPTION:** As more intelligent vehicles will ply the roads in the near future, a rapid increase of sensed environment data is anticipated. Information based on these acquired data needs to be extracted and shared in the most efficient way. To realize this, roadside units (RSUs) acting as hotspots and fog computing nodes should work together with vehicles in vehicular networks and intelligent transportation systems. In this paper, we consider a set of intersections in the city of Beijing as potential locations for strategically allocating fog computing hotspots to maximize the information shared among vehicles and fog nodes. Using empirical findings from mobility traces such as vehicular density, total daily number of transmissions, transmitted data size, and space mean speed, we propose the Information Sharing via Roadside unit Allocation (ISRA) strategy to determine the optimal locations for these fog computing hotspots. Simulation results show that for a given deployment limit, ISRA, when compared to three other conventional deployment schemes, is able to share on average 6%, 10% and 47% more road information with fewer packet transmissions (energy efficiency of 83%) in the vehicular network. In addition, ISRA is able to balance the information load among adjacent RSU fog nodes for better resource management.

**2.10 TITLE:** Controlling and filtering users data in Intelligent Transportation System

**AUTHOR:** Catalin Gosman, Tudor Cornea , Ciprian Dobre ,

**YEAR:** 2018

**DESCRIPTION:** GPS data can facilitate the construction of ITS services for route discovery, but in the same time malicious users can use the information in order to derive location patterns and geographical habits. Several ITS companies could gain interesting insights about the traffic and safety events, if they put together owned private data. However, at this moment, the security risks prohibit such an endeavor. In this paper, we illustrate a security model where ITS participants can specify how data sharing captured by an ITS application will behave in regards to their own privacy requirements. The proposed solution is able to mediate the differences between ITS applications needs regarding data usage under various context based constraints and user focused constraints defined using security policies for their shared data. The next topic discussed in the paper is our proposal of an appropriate ITS mechanism that manages to establish the level of trust in the information disseminated in the system. The trust level mechanism is used in order to decide whether and ITS event should be are advertised or not to other users or ITS applications.

**CHAPTER-3**

**SYSTEM ANALYSI**

**3.1 EXISTING SYSTEM**

Yolo based model is developed in the existing method to detect the vehicle number plate. Darknet has been utilized in the existing system for the training and testing process. Number of frame for the processing is higher than other deep learning approaches. A system is ran with a DSLR camera is used to capture an image of the vehicle using an Automatic Number plate Recognition (ANPR) to extract the license plate, and an Optical Character Recognition (OCR) is used to identify the characters of the plate which is then sent to the toll plaza via email. In existing system Optical Character Recognition (OCR) has been implemented. Threshold based vehicle plate number recognized in existing system

**3.2 DISADVANTAGES**

* Character not recognized
* Less accuracy
* Letter and character not recognized
  1. **PROPOSED SYSTEM**

The proposed system is designed in order to perform license plate recognition. In the proposed system vehicle license plate number is recognized using machine learning algorithm. ANPR performed in controlled lighting conditions with predictable license plate types can use basic image processing techniques. More advanced ANPR systems utilize dedicated object detectors, such as HOG + Linear SVM, Faster R-CNN, SSDs, and YOLO, to localize license plates in images. Suppose an ANPR system is mounted on a toll road. It needs to be able to detect the license plate of each car passing by, OCR the characters on the plate, and then store this information in a database so the owner of the vehicle can be billed for the toll.

Automatic License/Number Plate Recognition (ANPR/ALPR) is a process involving the following steps:

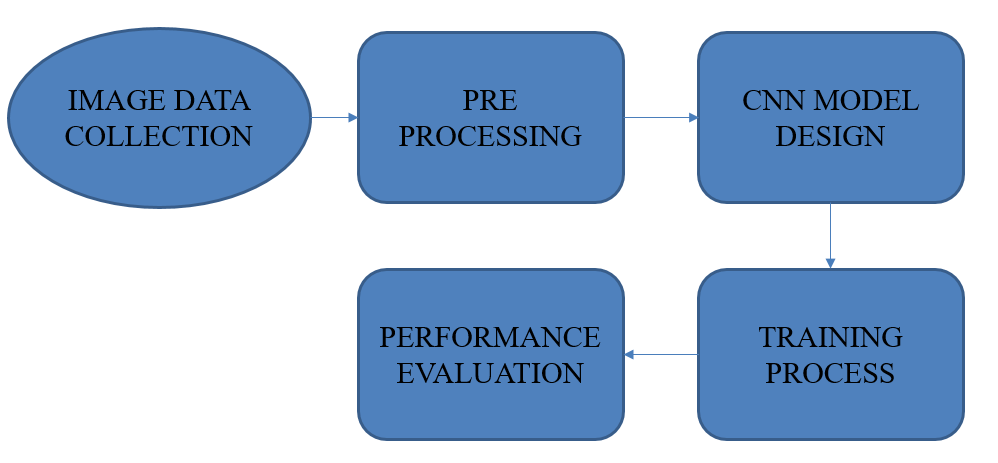
* **Step #1:** Detect and localize a license plate in an input image/frame
* **Step #2:** Extract the characters from the license plate
* **Step #3:** Apply some form of Optical Character Recognition (OCR) to recognize the extracted characters

**3.4 ADVANTAGE**

* Low detection time.
* Higher accuracy.

**3.5 SYSTEM ARCHITECTURE**

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.



**CHAPTER 4**

**MODULE IMPLEMENTATION**

**4.1 MODULE LIST**

* Data collection
* Image preprocessing
* Neural network design
* Model training
* Performance evaluation
  1. **MODULE IMPLEMENTATION**
     1. **DATA COLLECTION**
* Data collation is a process of collecting the number plate detected image dataset for our training process.
  + 1. **IMAGE PREPROCESSING**
* Image resizing, gray scale conversion and data augmentation are the preprocessing steps which are takeover in the Image Processing module.
  + 1. **NEURAL NETWORK DESIGN**
* The Design of conventional neural network is performed in this module.
* The Deep learning model has more than one layers for the number classification.
* ConV2D, Maxpooling, Batch normalization are the operations which are defined in the network.
  + 1. **MODEL TRAINING**
* After define the network, Model compiling and training with pre proposed data is take over in the module.

* + 1. **PERFORMANCE EVALUATION**
* The Performance evaluation is the process of calculate the performance metrics such as Accuracy and loss function of the both training and testing data.

**CHAPTER 5**

**SOFTWARE REQUIREMENTS**

**H/W SYSTEM CONFIGURATION:-**

* processor - Pentium – IV
* RAM - 4 GB (min)
* Hard Disk - 20 GB

**S/W SYSTEM CONFIGURATION:-**

* Operating System : Windows 7 or 8
* Software : Python Idle

**5.1 SOFTWARE ENVIRONMENT**

**Python Technology:**

**Python** is an interpreted, high-level, general-purpose programming language. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. **Python** is often described as a "batteries included" language due to its comprehensive standard library.

**Python Programing Language:**

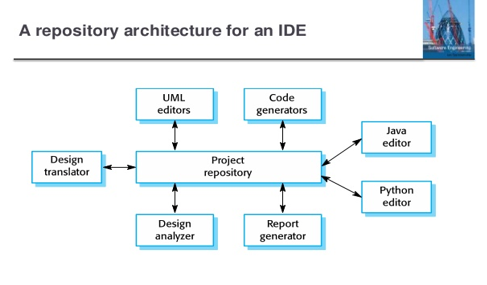
Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and met objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python packages with a wide range of functionality, including:

* Easy to Learn and Use
* Expressive Language
* Interpreted Language
* Cross-platform Language
* Free and Open Source
* Object-Oriented Language
* Extensible
* Large Standard Library
* GUI Programming Support
* Integrated

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

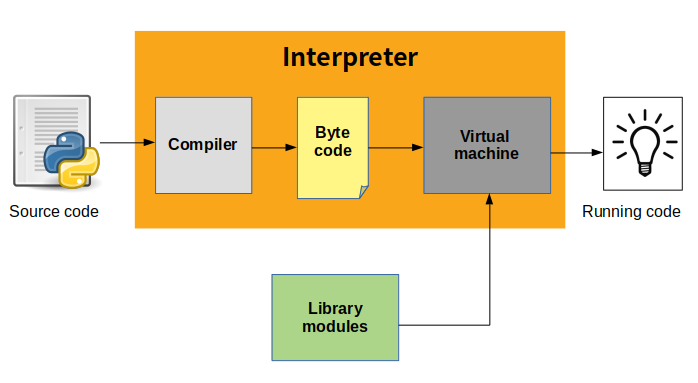
Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are optional. It has fewer syntactic exceptions and special cases than C or Pascal.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one and preferably only one obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Python is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar.

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.



**The Python Platform:**

The platform module in Python is used to access the underlying platform's data, such as, hardware, operating system, and interpreter version information. The platform module includes tools to see the platform's hardware, operating system, and interpreter version information where the program is running.

There are four functions for getting information about the current Python interpreter. python\_version() and python\_version\_tuple() return different forms of the interpreter version with major, minor, and patch level components. python\_compiler() reports on the compiler used to build the interpreter. And python\_build() gives a version string for the build of the interpreter.

Platform() returns string containing a general purpose platform identifier. The function accepts two optional Boolean arguments. If aliased is true, the names in the return value are converted from a formal name to their more common form. When terse is true, returns a minimal value with some parts dropped.

**What does python technology do?**

Python is quite popular among programmers, but the practice shows that business owners are also Python development believers and for good reason. Software developers love it for its straightforward syntax and reputation as one of the easiest programming languages to learn. Business owners or CTOs appreciate the fact that there’s a framework for pretty much anything – from web apps to machine learning.

Moreover, it is not just a language but more a technology platform that has come together through a gigantic collaboration from thousands of individual professional developers forming a huge and peculiar community of aficionados.

So what are the tangible benefits the language brings to those who decided to use it as a core technology? Below you will find just some of those reasons.

**PRODUCTIVITY AND SPEED**

It is a widespread theory within development circles that developing Python applications is approximately up to 10 times faster than developing the same application in Java or C/C++. The impressive benefit in terms of time saving can be explained by the clean object-oriented design, enhanced process control capabilities, and strong integration and text processing capacities. Moreover, its own unit testing framework contributes substantially to its speed and productivity.

**PYTHON IS POPULAR FOR WEB APPS**

Web development shows no signs of slowing down, so technologies for rapid and productive web development still prevail within the market. Along with JavaScript and Ruby, Python, with its most popular web framework Django, has great support for building web apps and is rather popular within the web development community.

**OPEN-SOURCE AND FRIENDLY COMMUNITY**

As stated on the official website, it is developed under an OSI-approved open source license, making it freely usable and distributable. Additionally, the development is driven by the community, actively participating and organizing conference, meet-ups, hackathons, etc. fostering friendliness and knowledge-sharing.

**PYTHON IS QUICK TO LEARN**

It is said that the language is relatively simple so you can get pretty quick results without actually wasting too much time on constant improvements and digging into the complex engineering insights of the technology. Even though Python programmers are really in high demand these days, its friendliness and attractiveness only help to increase number of those eager to master this programming language.

**BROAD APPLICATION**

It is used for the broadest spectrum of activities and applications for nearly all possible industries. It ranges from simple automation tasks to gaming, web development, and even complex enterprise systems. These are the areas where this technology is still the king with no or little competence:

* Machine learning as it has a plethora of libraries implementing machine learning algorithms.
* Web development as it provides back end for a website or an app.
* Cloud computing as Python is also known to be among one of the most popular cloud-enabled languages even used by Google in numerous enterprise-level software apps.
* Scripting.
* Desktop GUI applications.

**Python compiler**

The Python compiler package is a tool for analyzing Python source code and generating Python bytecode. The compiler contains libraries to generate an abstract syntax tree from Python source code and to generate Python bytecode from the tree.

The compiler package is a Python source to bytecode translator written in Python. It uses the built-in parser and standard parser module to generate a concrete syntax tree. This tree is used to generate an abstract syntax tree (AST) and then Python bytecode.

The full functionality of the package duplicates the built-in compiler provided with the Python interpreter. It is intended to match its behavior almost exactly. Why implement another compiler that does the same thing? The package is useful for a variety of purposes. It can be modified more easily than the built-in compiler. The AST it generates is useful for analyzing Python source code.

**The basic interface**

The top-level of the package defines four functions. If you import compiler, you will get these functions and a collection of modules contained in the package.

**compiler.parse(buf)**

Returns an abstract syntax tree for the Python source code in buf. The function raises Syntax Error if there is an error in the source code. The return value is a compiler.ast. Module instance that contains the tree.

**compiler.parseFile(path)**

Return an abstract syntax tree for the Python source code in the file specified by path. It is equivalent to parse(open(path).read()).

**LIMITATIONS**

There are some problems with the error checking of the compiler package. The interpreter detects syntax errors in two distinct phases. One set of errors is detected by the interpreter’s parser, the other set by the compiler. The compiler package relies on the interpreter’s parser, so it get the first phases of error checking for free. It implements the second phase itself, and that implementation is incomplete. For example, the compiler package does not raise an error if a name appears more than once in an argument list: def f(x, x): ...

A future version of the compiler should fix these problems.

**PYTHON ABSTRACT SYNTAX**

The compiler.ast module defines an abstract syntax for Python. In the abstract syntax tree, each node represents a syntactic construct. The root of the tree is Module object.

The abstract syntax offers a higher level interface to parsed Python source code. The parser module and the compiler written in C for the Python interpreter use a concrete syntax tree. The concrete syntax is tied closely to the grammar description used for the Python parser. Instead of a single node for a construct, there are often several levels of nested nodes that are introduced by Python’s precedence rules.

The abstract syntax tree is created by the compiler.transformer module. The transformer relies on the built-in Python parser to generate a concrete syntax tree. It generates an abstract syntax tree from the concrete tree.

The transformer module was created by Greg Stein and Bill Tutt for an experimental Python-to-C compiler. The current version contains a number of modifications and improvements, but the basic form of the abstract syntax and of the transformer are due to Stein and Tutt.

**AST NODES**

The compiler.ast module is generated from a text file that describes each node type and its elements. Each node type is represented as a class that inherits from the abstract base class compiler.ast.Node and defines a set of named attributes for child nodes

class compiler.ast.Node

The Node instances are created automatically by the parser generator. The recommended interface for specific Node instances is to use the public attributes to access child nodes. A public attribute may be bound to a single node or to a sequence of nodes, depending on the Node type. For example, the bases attribute of the Class node, is bound to a list of base class nodes, and the doc attribute is bound to a single node.

Each Node instance has a lineno attribute which may be None. XXX Not sure what the rules are for which nodes will have a useful lineno.

**All Node objects offer the following methods:**

**getChildren()**

Returns a flattened list of the child nodes and objects in the order they occur. Specifically, the order of the nodes is the order in which they appear in the Python grammar. Not all of the children are Node instances. The names of functions and classes, for example, are plain strings.

**getChildNodes()**

Returns a flattened list of the child nodes in the order they occur. This method is like getChildren(), except that it only returns those children that are Node instances.

The While node has three attributes: test, body, and else\_. (If the natural name for an attribute is also a Python reserved word, it can’t be used as an attribute name. An underscore is appended to the word to make it a legal identifier, hence else\_ instead of else.)

The if statement is more complicated because it can include several tests.

The If node only defines two attributes: tests and else\_. The tests attribute is a sequence of test expression, consequent body pairs. There is one pair for each if/elif clause. The first element of the pair is the test expression. The second elements is a Stmt node that contains the code to execute if the test is true.

The getChildren() method of If returns a flat list of child nodes. If there are three if/elif clauses and no else clause, then getChildren() will return a list of six elements: the first test expression, the first Stmt, the second text expression, etc.

The following table lists each of the Node subclasses defined in compiler.ast and each of the public attributes available on their instances. The values of most of the attributes are themselves Node instances or sequences of instances. When the value is something other than an instance, the type is noted in the comment. The attributes are listed in the order in which they are returned by getChildren() and getChildNodes().

**DEVELOPMENT ENVIRONMENTS:**

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which the user enters statements sequentially and receives results immediately.

Other shells, including IDLE and IPython, add further abilities such as auto-completion, session state retention and syntax highlighting.

**IMPLEMENTATIONS**

**Reference implementation**

CPython is the reference implementation of Python. It is written in C, meeting the C89 standard with several select C99 features. It compiles Python programs into an intermediate bytecode which is then executed by its virtual machine. CPython is distributed with a large standard library written in a mixture of C and native Python. It is available for many platforms, including Windows and most modern Unix-like systems. Platform portability was one of its earliest priorities.

**Other implementations**

PyPy is a fast, compliant interpreter of Python 2.7 and 3.5. Its just-in-time compiler brings a significant speed improvement over CPython but several libraries written in C cannot be used with it.

Stackless Python is a significant fork of CPython that implements microthreads; it does not use the C memory stack, thus allowing massively concurrent programs. PyPy also has a stackless version.

MicroPython and CircuitPython are Python 3 variants optimized for microcontrollers. This includes Lego Mindstorms EV3.

RustPython is a Python 3 interpreter written in Rust.

**Unsupported implementations**

Other just-in-time Python compilers have been developed, but are now unsupported:

Google began a project named Unladen Swallow in 2009, with the aim of speeding up the Python interpreter five-fold by using the LLVM, and of improving its multithreading ability to scale to thousands of cores, while ordinary implementations suffer from the global interpreter lock.

Psyco is a just-in-time specialising compiler that integrates with CPython and transforms bytecode to machine code at runtime. The emitted code is specialized for certain data types and is faster than standard Python code.

In 2005, Nokia released a Python interpreter for the Series 60 mobile phones named PyS60. It includes many of the modules from the CPython implementations and some additional modules to integrate with the Symbian operating system. The project has been kept up-to-date to run on all variants of the S60 platform, and several third-party modules are available. The Nokia N900 also supports Python with GTK widget libraries, enabling programs to be written and run on the target device.

**Cross-compilers to other languages**

There are several compilers to high-level object languages, with either unrestricted Python, a restricted subset of Python, or a language similar to Python as the source language:

* Jython enables the use of the Java class library from a Python program.
* IronPython follows a similar approach in order to run Python programs on the .NET Common Language Runtime.
* The RPython language can be compiled to C, and is used to build the PyPy interpreter of Python.
* Pyjs compiles Python to JavaScript.
* Cython compiles Python to C and C++.
* Numba uses LLVM to compile Python to machine code.
* Pythran compiles Python to C++.
* Somewhat dated Pyrex (latest release in 2010) and Shed Skin (latest release in 2013) compile to C and C++ respectively.
* Google's Grumpy compiles Python to Go.
* MyHDL compiles Python to VHDL.
* Nuitka compiles Python into C++.

**PERFORMANCE**

A performance comparison of various Python implementations on a non-numerical (combinatorial) workload was presented at EuroSciPy '13.

**API DOCUMENTATION GENERATORS**

Python API documentation generators include:

* Sphinx
* Epydoc
* HeaderDoc
* Pydoc

**USES**

Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP, Inkscape, Scribus and Paint Shop Pro, and musical notation programs like scorewriter and capella. GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS. It has also been used in several video games, and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.

Python is commonly used in artificial intelligence projects with the help of libraries like TensorFlow, Keras and Scikit-learn. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing.

Many operating systems include Python as a standard component. It ships with most Linux distributions, AmigaOS 4, FreeBSD (as a package), NetBSD, OpenBSD (as a package) and macOS and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage.

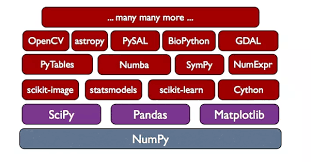
Python is used extensively in the information security industry, including in exploit development.

Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python. The Raspberry Pi single-board computer project has adopted Python as its main user-programming language.

LibreOffice includes Python, and intends to replace Java with Python. Its Python Scripting Provider is a core feature since Version 4.0 from 7 February 2013.

**PANDAS**

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

**Library features**

* Data Frame object for data manipulation with integrated indexing.
* Tools for reading and writing data between in-memory data structures and different file formats.
* Data alignment and integrated handling of missing data.
* Reshaping and pivoting of data sets.
* Label-based slicing, fancy indexing, and sub setting of large data sets.
* Data structure column insertion and deletion.
* Group by engine allowing split-apply-combine operations on data sets.
* Data set merging and joining.
* Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
* Time series-functionality: Date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
* Provides data filtration.

**CHAPTER 6**

**SYSTEM DESIGN**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

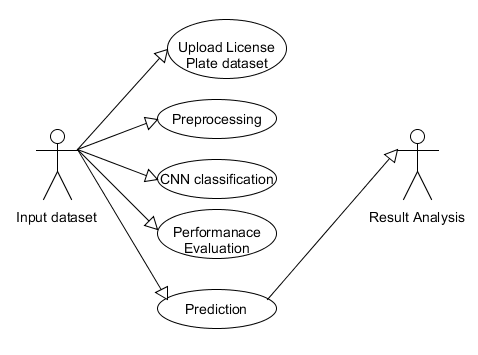
**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

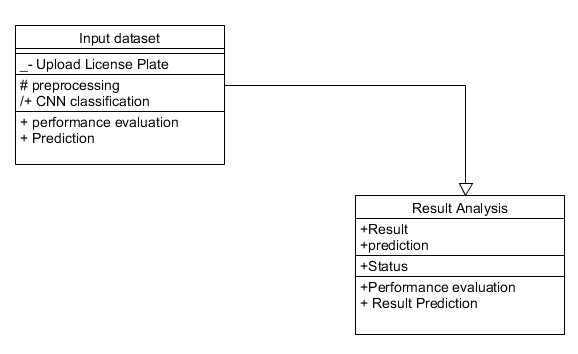
**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



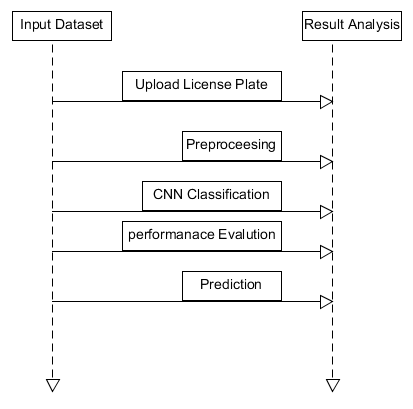
**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



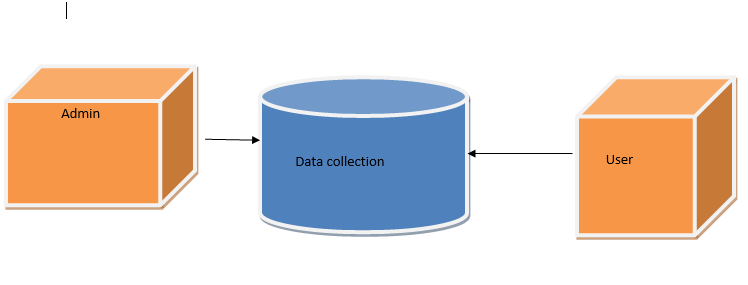
**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**DEPLOYMENT**

Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware. UML is mainly designed to focus on the software artifacts of a system. However, these two diagrams are special diagrams used to focus on software and hardware components.



**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

DATASET

COLLECTION

PREPROCESSING

DATA

CNN

ALGORITHM

RESULT

DATA

BASE

TRAINED FILE

**CHAPTER 7**

**SYSTEM TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

***System Test***

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

***White Box Testing***

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**7.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

**7.2 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**7.3 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**CHAPTER 8**

**CONCLUSION**

This paper implemented two methods of license plate detection, one is based on Sobel edge detection and the other is based on morphological gradient detection. Through the comparison of the aspects of license plate detection accuracy and license plate tilt detection rate, we can conclude that the license plate detection based on Sobel edge detection is suitable for the vehicle images whose camera shooting direction is parallel to the license plate area and the license plate detection based on morphological gradient detections is suitable for detecting the vehicle image with a certain tilt in the license plate position. Therefore, for different types of vehicle images, we can use different license plate detection methods to improve the detection accuracy.

**REFERENCE**

[1] Hsieh, Jun-Wei, Shih-Hao Yu, and Yung-Sheng Chen. "Morphology- based license plate detection from complex scenes." Object recognition is supported by user interaction for service robots. Vol. 3. IEEE, 2002.

[2] Jin, Lisheng, et al. "License plate recognition algorithm for passenger cars in Chinese residential areas." Sensors 12.6 (2012): 8355-8370.

[3] Yan, Qing. "Method of license plate location based on license plate texture and HSV color space." Information Engineering and Applications. Springer, London, 2012. 962-970.

[4] Gao, Wenshuo, et al. "An improved Sobel edge detection." 2010 3rd International conference on computer science and information technology. Vol. 5. IEEE, 2010.

[5] Deng, Caixia, Weifeng Ma, and Yin Yin. "An edge detection approach of image fusion based on improved Sobel operator." 2011 4th International Congress on Image and Signal Processing. Vol. 3. IEEE, 2011.

[6] Neubeck, Alexander, and Luc Van Gool. "Efficient non-maximum suppression." 18th International Conference on Pattern Recognition (ICPR'06). Vol. 3. IEEE, 2006.

[7] Chaudhuri, D., and Ashok Samal. "A simple method for fitting of bounding rectangle to closed regions." Pattern recognition 40.7 (2007): 1981-1989.

[8] Rivest, Jean-Francois, Pierre Soille, and Serge Beucher. "Morphological gradients." Journal of Electronic Imaging 2.4 (1993): 326-337.

[9] Evans, Adrian N., and Xin U. Liu. "A morphological gradient approach to color edge detection." IEEE Transactions on Image Processing 15.6 (2006): 1454-1463.

[10] Leavers, V. F. "Which hough transform?." CVGIP: Image understanding 58.2 (1993): 250-264.

[11] Dunteman, George H. Principal components analysis. No. 69. Sage, 1989.

[12] Axelsson, Owe. "A generalized conjugate gradient, least square method." Numerische Mathematik 51.2 (1987): 209-227.

[13] Anagnostopoulos, Christos Nikolaos E., et al. "A license plate-recognition algorithm for intelligent transportation system applications." IEEE Transactions on Intelligent transportation systems 7.3 (2006): 377-392.

[14] Du, Shan, et al. "Automatic license plate recognition (ALPR): A state-of-the-art review." IEEE Transactions on circuits and systems for video technology 23.2 (2012): 311-325.