VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI

A PROJECT REPORT ON

"AUTOMATIC LICENSE PLATE DETECTION AND RECOGNITION"

Submitted in partial fulfillment for the award of Degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE &ENGINEERING

By

HEMALATHA S	**4AL17CS033**
JAYALAKSHMI M	**4AL17CS035**
LATHIK MOGER	**4AL17CS045**

Under the Guidance of

Mr. Hemanth Kumar N P

Senior Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MOODBIDRI-574225, KARNATAKA

2020 – 2021

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MIJAR, MOODBIDRI D.K. -574225, KARNATAKA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"AUTOMATIC LICENSE PLATE DETECTION AND RECOGNITON USING DEEP LEARNING"** has been successfully completed by

HEMALATHA S	**4AL17CS033**
JAYALAKSHMI M	**4AL17CS035**
LATHIK MOGER	**4AL17CS045**

the bonafide students of **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING, ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the year 2020–2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the Bachelor of Engineering Degree.

Mr. Hemanth Kumar N P **Project Guide**	**Dr. Manjunath Kotari** **Head Of The Department**	**Dr. Peter Fernandes** **Prinicipal**

External Viva

Name of the Examiners	**Signature with Date**
1.	
2.	

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MIJAR, MOODBIDRI D.K. -574225, KARNATAKA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DECLARATION

We,

HEMALATHA S

JAYALAKSHMI M

LATHIK MOGER

hereby declare that the dissertation entitled **"AUTOMATIC LICENSE PLATE DETECTION AND RECOGNITION USING DEEP LEARNING"** is completed and written by us under the supervision of our guide **Mr. Hemanth Kumar N P**, Assistant Professor, Department of Computer and Engineering, Alva's Institute of Engineering and Technology, Moodbidri, in partial fulfillment of requirements for the award of the degree **BACHELOR OF ENGINEERING** in **DEPARTMENT OF COMPUTER AND ENGINEERING** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAVI** during the **academic year 2020 – 2021. The dissertation report is original and it has not been submitted for** any other degree in any university.

HEMALATHA S	**4AL17CS033**
JAYALAKSHMI M	**4AL17CS035**
LATHIK MOGER	**4AL17CS045**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, with gratitude we acknowledge all those whose guidance and encouragement served as beacon of light and crowned the effort with success.

We thank our project guide **Mr. Hemanth Kumar N P,** Senior Assistant Professor in Department of Computer Science & Engineering, who has been our source of inspiration. He has been especially enthusiastic in giving his valuable guidance and critical reviews.

The selection of this project work as well as the timely completion is mainly due to the interest and persuasion of my project coordinator **Mr. Hemanth Kumar N P,** Senior Assistant Professor, Department of Computer Science & Engineering. We will remember her contribution for ever.

We sincerely thank, **Dr. Manjunath Kotari**, Professor and Head, Department of Computer Science & Engineering who has been the constant driving force behind the completion of the project.

We thank Principal **Dr. Peter Fernandes,** for his constant help and support throughout.

We are also indebted to **Management of Alva's Institute of Engineering and Technology, Mijar, Moodbidri** for providing an environment which helped us in completing the project.

Also, we thank all the teaching and non-teaching staff of Department of Computer Science & Engineering for the help rendered.

Finally we would like to thank my parents and friends whose encouragement and support wasinvaluable.

HEMALATHA S	4AL17CS033
JAYALAKSHMI M	4AL17CS035
LATHIK MOGER	4AL17CS045

ABSTRACT

Recognition of cars is very important for the control and surveillance systems. Automobiles can be recognized by number plates, which contains a unique combination of alphabets and numbers. However, it's a hard and intensive job for humans to manually recognize all the parked or passing car number plates. In this paper, we approach a training-based pathway for vehicle number plate recognition. Most of the previous works in automatic number plate recognition (ANPR) systems have limitations in their working conditions, like for example restricting them to stationary backgrounds, indoor area, restricted vehicle speeds, prescribed driveways, fixed illumination, or match the predefined distance between camera and vehicle. The main objective of our work is to create a robust number plate recognition model that works under different illuminations and angles. We created our recognition model by training on our manually collected car number plate dataset using YOLO-V3. The algorithm has been tested over 640 images which are of different colors, and illuminations. In this project, an efficient approach has been proposed to localize every clearly visible object from an image. For object detection we have processed every input image to overcome several complexities, to achieve better result we use object detection algorithm. We will also implement Convolution Neural Network based on object detection model i.e YOLO-V3. Finally, single character within the registration code is detected. The aim is to indicate that the planned technique achieved high accuracy by optimizing numerous parameters that has higher recognition rate than the standard ways.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Computer vision acts as a tool to perceive and grasp knowledge. The three basic works undertaken by a computer vision algorithm include classification, detection and localization. These three fundamental tasks provide both opportunities and challenges to the field of computer vision. They have started seeking the attention of researches and there came a tremendous increase in the work over these fields. One reason for this tremendous growth is the integration of deep convolutional neural network to the field of computer vision. Image classification intends to convert an image to a label. The process of object detection, another important task in computer vision, points to detect the presence of entire objects within an image. For example, driverless cars otherwise called as self-driving cars are a couple that use a combination of software and sensors, not only to detect the presence of other cars but also the presence of trees, human beings, animals, other vehicles etc. on the road. The third main task in computer vision is called object localization. The localization task is similar to the task of detection.

Even though now that we know that Automatic Number Plate Recognition(ANPR) is widely known but not used because of its usage of the old technology, we are proposing to develop an application to manage the visitors in the residential areas using the same technology with GPU centric algorithms. Now to train the system we are going to use GPU centric algorithms like YOLO to compete with different challenges we face while working with ANPR in countries like India. In India we face challenges like, the cameras used are mostly used for surveillance purposes and not for training and testing of deep learning models, the second challenge we face is that the number plate sizes and character patterns vary from different vehicles which really makes things complicated. But, by using GPU centric algorithms and using some good camera handling techniques we can achieve good accuracy to implement different applications like visitor management systems using computer vision technologies like Automatic Number Plate Recognition.

Object detection deals with detection of a variable number of object classes whereas localization deals with the identification of a fixed number of object classes. The localization task is considered to be a superset of the computer vision task of image classification. With this, we are not only required to know whether there is an object in the image of the given class but also to know where exactly the object is in the image. Object detection is regarded as the first process in most computer vision systems. It has several applications and some of them include security systems, human-computer interactions, robotics, product detection in manufacturing industries, people counting and so on. From the last decade, the field of machine learning has started to move forward in a purposeful way, as a result, the researches have started to concentrate more over these techniques especially for the matter of object detection. Machine learning-based object detection consists of two phases, a training phase as well as a testing phase. An extensively large number of images are used for the purpose of training. Increasing the number of images enables the computer to learn the class of objects correctly and helps to properly identify the objects belonging to different classes. Testing phase involves the

task of testing whether the machine responds correctly by giving different inputs or test cases. By giving an image to a computer it tries to learn every bit of object features within the image by a process called feature extraction.

1.1 DEEP LEARNING

Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance. Artificial neural networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analog.

The adjective "deep" in deep learning comes from the use of multiple layers in the network. Work showed that a linear perceptron cannot be a universal classifier, and then that a network with a non-polynomial activation function with one hidden layer of unbounded width can on the other hand so be. Deep learning is a modern variation which is concerned with an unbounded number of layers of bounded size, which permits practical application and optimized implementation, while retaining theoretical universality under mild conditions. In deep learning the layers are also permitted to be heterogeneous and to deviate widely from biologically informed connectionist models, for the sake of efficiency, trainability and understandability, whence the "structured" part. While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:

- Deep learning requires large amounts of labeled data. For example, driverless car development requires millions of images and thousands of hours of video.
- Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

Deep learning applications are used in industries from automated driving to medical devices.

- Automated Driving: Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.
- Aerospace and Defense: Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.
- Medical Research: Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.
- Industrial Automation: Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.
- Electronics: Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

Most deep learning methods use neural network architectures, deep learning models are often referred to as deep neural networks. The term “deep” usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.

Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction. One of the most popular types of deep neural networks is known as convolutional neural networks (CNN or ConvNet). A CNN convolves learned features with input data, and uses 2D convolutional layers, making this architecture well suited to processing 2D data, such as images.

CNNs eliminate the need for manual feature extraction, so you do not need to identify features used to classify images. The CNN works by extracting features directly from images. The relevant features are not pretrained; they are learned while the network trains on a collection of images. This automated feature extraction makes deep learning models highly accurate for computer vision tasks such as object classification.

1.2 DEEP LEARNING VS MACHINE LEARNING

The easiest takeaway for understanding the difference between machine learning and deep learning is to know that deep learning *is* machine learning.More specifically, deep learning is considered an evolution of machine learning. It uses a programmable neural network that enables machines to make accurate decisions without help from humans.Deep learning is just a subset of machine learning. In fact, deep learning

technically is machine learning and functions in a similar way (hence why the terms are sometimes loosely interchanged).

While basic machine learning models do become progressively better at whatever their function is, they still need some guidance. If an AI algorithm returns an inaccurate prediction, then an engineer has to step in and make adjustments. With a deep learning model, an algorithm can determine on its own if a prediction is accurate or not through its own neural network. In the flashlight example: it could be programmed to turn on when it recognizes the audible cue of someone saying the word *"dark"*. As it continues learning, it might eventually turn on with any phrase containing that word. Now if the flashlight had a deep learning model, it could figure out that it should turn on with the cues "I can't see*"* or *"*the light switch won't work," perhaps in tandem with a light sensor. A deep learning model is able to learn through its own method of computing- a technique that makes it seem like it has its own brain.

Machine learning offers a variety of techniques and models you can choose based on your application, the size of data you're processing, and the type of problem you want to solve. A successful deep learning application requires a very large amount of data (thousands of images) to train the model, as well as GPUs, or graphics processing units, to rapidly process your data.

When choosing between machine learning and deep learning, consider whether you have a high-performance GPU and lots of labeled data. If you don't have either of those things, it may make more sense to use machine learning instead of deep learning. Deep learning is generally more complex, so you'll need at least a few thousand images to get reliable results. Having a high-performance GPU means the model will take less time to analyze all those images.

1.3 CONVOLUTIONAL NEURAL NETWORK(CNN)

A convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant. CNNs are regularized versions of multilayer perceptron. Multilayer perceptron usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme. Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward

neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images.

Deep learning is a subfield of machine learning, which is, in turn, a subfield of artificial intelligence (AI) as shown in Figure 1.1. The central goal of AI is to provide a set of algorithms and techniques that can be used to solve problems that humans perform intuitively and near automatically, but are otherwise very challenging for computers. A great example of such a class of AI problems is interpreting and understanding the contents of an image – this task is something that a human can do with little-to-no effort, but it has proven to be extremely difficult for machines to accomplish.

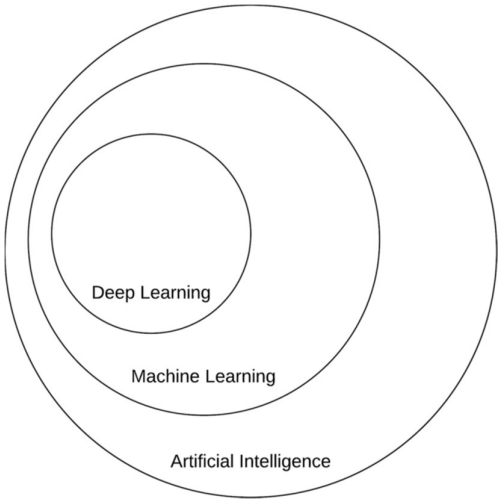

Figure 1.1: A Venn diagram describing deep learning as a subfield of machine learning which a subfield of artificial intelligence is in turn.

While AI embodies a large, diverse set of work related to automatic machine reasoning (inference, planning, heuristics, etc.), the machine learning subfield tends to be specifically interested in pattern recognition and learning from data.

Artificial Neural Networks (ANNs) are a class of machine learning algorithms that learn from data and specialize in pattern recognition, inspired by the structure and function of the brain. As we'll find out, deep learning belongs to the family of ANN algorithms, and in most cases, the two terms can be used interchangeably. In fact, you may be surprised to learn that the deep learning field has been around for over 60 years, going by different names and incarnations based on research trends, available hardware and datasets, and popular options of prominent researchers at the time.
we'll review a brief history of deep learning, discuss what makes a neural network "deep", and discover the concept of "hierarchical learning" and how it has made deep learning one of the major success stories in modern day machine learning and computer vision.

A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for each neuron in the second layer as shown in Figure 1.2. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. By using regularized weights over fewer parameters, the vanishing gradient and exploding gradient problems seen during backpropagation in traditional neural networks are avoided. Pre-trained models are usually shared in the form of the millions of parameters/weights the model achieved while being trained to a stable state.

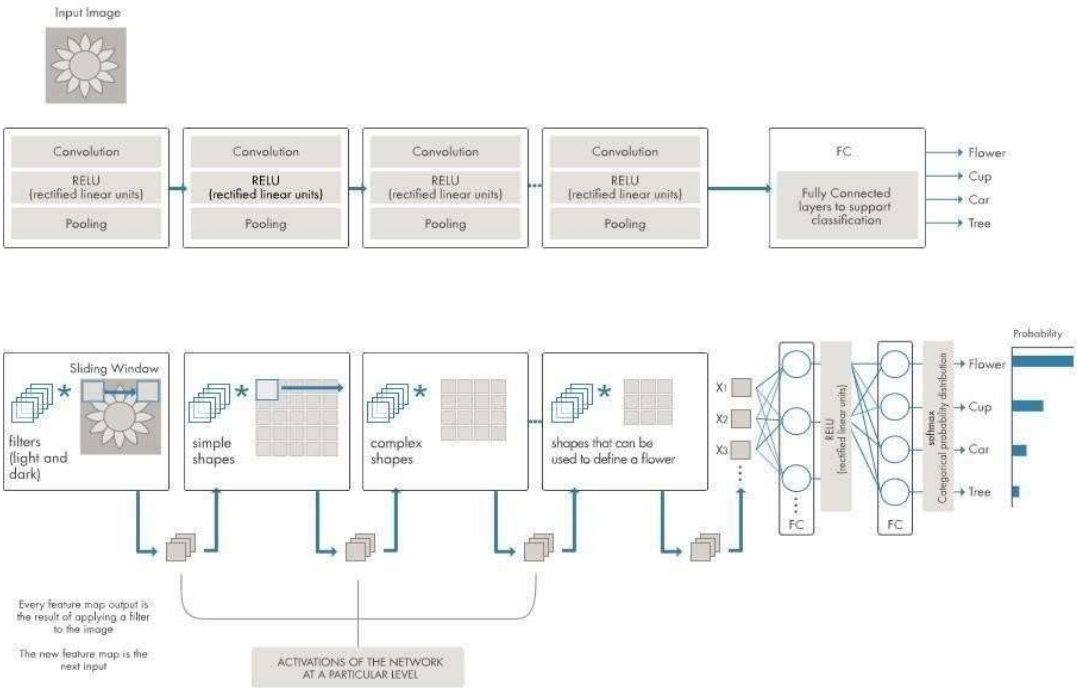

Figure 1.2: Layers of CNN

There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters.

- Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as "pattern matching", "pattern recognition", or "image correlation". This relies on the input glyph being correctly isolated from the rest of the image, and on the stored glyph being in a similar font and at the same scale. This

technique works best with typewritten text and does not work well when new fonts are encountered. This is the technique the early physical photocell-based OCR implemented, rather directly.

- Feature extraction decomposes glyphs into "features" like lines, closed loops, line direction, and line intersections. The extraction features reduce the dimensionality of the representation and makes the recognition process computationally efficient. These features are compared with an abstract vector-like representation of a character, which might reduce to one or more glyph prototypes. General techniques of feature detection in computer vision are applicable to this type of OCR, which is commonly seen in "intelligent" handwriting recognition and indeed most modern OCR software. Nearest neighbor classifiers such as the k-nearest neighbors algorithm are used to compare image features with stored glyph features and choose the nearest match.

Software such as Cuneiform and Tesseract use a two-pass approach to character recognition. The second pass is known as "adaptive recognition" and uses the letter shapes recognized with high confidence on the first pass to recognize better the remaining letters on the second pass. This is advantageous for unusual fonts or low-quality scans where the font is distorted (e.g. blurred or faded).

Modern OCR software like for example OCRopus or Tesseract uses neural networks which were trained to recognize whole lines of text instead of focusing on single characters. A new technique known as iterative OCR automatically crops a document into sections based on page layout. OCR is performed on the sections individually using variable character confidence level thresholds to maximize page-level OCR accuracy. The OCR result can be stored in the standardized ALTO format, a dedicated XML schema maintained by the United States Library of Congress. Other common formats include hOCR and PAGE XML. For a list of optical character recognition software see Comparison of optical character recognition software.

Post-processing: OCR accuracy can be increased if the output is constrained by a lexicon – a list of words that are allowed to occur in a document. This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains words not in the lexicon, like proper nouns. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy. The output stream may be a plain text stream or file of characters, but more sophisticated OCR systems can preserve the original layout of the page and produce, for example, an annotated PDF that includes both the original image of the page and a searchable textual representation. "Near-neighbor analysis" can make use of co-occurrence frequencies to correct errors, by noting that certain words are often seen together. For example, "Washington, D.C." is generally far more common in English than "Washington DOC". Knowledge of the grammar of the language being scanned can also help determine if a word is likely to be a verb or a noun, for example, allowing greater accuracy. The Levenshtein Distance algorithm has also been used in OCR post-processing to further optimize results from an OCR API.

1.4 PROBLEM STATEMENT

- In any object recognition system, there are two major problems that need to be solved Detecting an object, and Recognizing that object.
- The changing weather conditions, are what make the field of LPR a good candidate for testing Pattern Recognition techniques.
- License plates come in different sizes, different Width-Height ratios, different color, the fonts used for digits on license plates are not the same for all license plates.
- The existing system uses MATLAB, Open CV and TensorFlow to extract the contents from the license plate

1.5 OBJECTIVES

- To overcome the computational expense and to provide more accurate bounding boxes, has introduced the YOLO (You Only Look Once) method.
- Identification of fruits is by using YOLO-V3 algorithm.
- We proposed a solution to help the people to determine the type of apple more accurately and to predict the type of images of apple.
- The quantity and the quality of the images have increased, it leads to both opportunities and challenges in this field.
- Images are high quality images taken from the top of the earth at varying heights, at varying light conditions for variety of applications. The accuracy of the object detection results can be improved by increasing the count of the images.

Getting the best results in deep learning requires experimenting with different values for training parameters, an important step called hyper parameter tuning. Since Deep Learning Pipelines enables exposing deep learning training as a step in Spark's machine learning pipelines, users can rely on the hyper parameter tuning infrastructure already built into Spark. Pre-trained models are usually shared in the form of the millions of parameters, weights the model achieved while being trained to a stable state. Pre-trained models are available for everyone to use through different means. The famous deep learning Python library, keras, provides an interface to download some popular models.

CHAPTER 2

LITERATURE REVIEW

Some protocol developed previously will be discussed in this section. A significant amount of work has been done over the last couple of years on image processing technique and deep learning for object detection purpose. Several different recognition and detection algorithms for vehicle reconnaissance have evolved in this field. We can see different current techniques occurring from literature review.

[1] K.K. Kim built a license plate recognition system by following a learning protocol. The camera captures an image inside the car detection module. Then, the picture of the candidate region is provided as output. The two TDNNs were taken as the horizontal and vertical filters to find the license plate. The segmentation rate was 97.5 per cent, and a recognition rate of 97.2 per cent for the proposed system.

[2] Chin-Chuan Han suggested a device that not only tracks several targets but also obtains high- quality images on plate numbers. A computer with a tuned dual-camera system has been built here by the author; a stationary camera and a pan-tilt-zoom camera are designed to monitor moving conveyance in an open field. The license plate for recognition has been sequentially identified by CNN classifier. As 64 vehicles entered this region illegally, data was composed manually from the science pictures and 59 IDs were accurately detected using this tool .

[3] Madhusree Mondal in 2017 developed an ANPR framework focused on the learning capabilities of convolutional neural networks. The self-synthesized function of CNN was used here, as it distinguishes the vehicle states from the number plate. The system was organized in this work in an echelon network of feature detectors that conducted consecutive processing of visual data pertaining to the dominant visual processing experience of the visual cortex, which influenced the computational model of the CNN. The findings of this research were observed with fewer training samples and turned out to be as 90 per cent higher precision rate.

[4] Andrew S. Agbemenu in 2018 proposed an ANPR method based on the characteristics and variations of the plates therein. The author has proposed in this work an algorithm that is enhanced to perform with Ghanaian license plate for conveyance. The designed model used two candidate detection algorithms as the detection of edges and the algorithms matching the template. The device then implemented the character segmentation technique particularly with square plates to prevent noise effects, arrangement of characters and skewing. At the final point, character recognition was rendered with the use of tesseract OCR engine. Feature detection was slightly low but had a good success rate, with an average speed of 0.185s detecting 454 plates with 90.8 per cent accuracy. The optical character recognition provided an average of 0.031s for the procedure and successfully identified approximately 60 per cent of the detected plates.

[5] Rayson Laroca in 2018 proposed an ALPR system which discussed the robustness and effectiveness of a framework based on the state-of the- art YOLO artifact detector. The CNN are qualified and adapted for each ALPR stage to be resilient

under different conditions. In this work, the author developed a two-stage attempt explicitly for the segmentation and identification of characters, using simple data augmentation artifice such as inverted number plates and character returns. The findings for the UFPR-ALPR dataset were found to be difficult as both commercial systems were below recognition levels of 70 per cent. Yet the result was higher with a recognition rate of 78.33 per cent for the proposed system.

[6] Prashengit Dhar in 2018 developed an automated LPR program to support ITS for the identification of Bangladeshi license plates. This work plate shows clearly white background with black fonts. Prewitt operators performed the detection of the number plate to segment the edges. Morphological dilation was performed to accentuate the points. Eventually, deep CNN was used to accomplish the reconnaissance job. In character classification, the protocol showed a strong precision rate of 99.6 per cent.

[7] Cheng-Hung Lin in 2019 proposed a three-stage license plate recognition system based on Mask-RCNN which was used for various shooting angles and numerous oblique images. The author used YOLOv2 for the associated conveyance in the preceding stage for vehicle detection. Next stage was the location of the license plate where YOLOv2 was again performed to detect number plate. During this phase, YOLOv2 separates the images of phase I captured vehicles into 19 x 19 grids. In the final step, the author used Mask R-CNN for character recognition. The results in this work depicts that the proposed model could classify vehicle number plates including bevel angles above 0-60 degrees and further accomplished the mAP rating of around 91 per cent.

[8] Nazmus Saif in 2019 have proposed a system to detect and recognize the Bangla license plate from the vehicle picture by using the convolutional neural networks. In this work, main focus to choose convolution neural network in the designed system is preferred because of its configuration for the end-to-end pipeline. CNN clearly outperformed conventional image processing algorithms for their case, and compared generalized CNN models better in different scenarios. The detection research was done using YOLOv3 which consists of 53 convolutional model layers. The second stage after identification is image segmentation and recognition of the characters it is. During this step, the device whips out the number plate region and then moves it to the second YOLO model for segmentation and platform image recognition. As a result, the model was checked with 200 images and correctly recognized the license plate number for 199 images, i.e. 99.5 per cent accuracy rate.

CHAPTER 3

SYSTEM REQUIREMENTS AND SPECIFICATION

Computer Aided learning is a rapidly growing dynamic area of research in Autonomous vehicle industry. The recent researchers in machine learning and Artificial intelligence promise the improved accuracy of perception of Emotion and drowsiness detection. Here the computers are enabled to think by developing intelligence by learning. There are many types of Machine Learning Techniques and which are used to classify the data sets.

3.1 FUNCTIONAL REQUIREMENTS

The proposed application should be able to identify input face emotion and detect emotion type. Functional requirements define a function of a system and its components. A function is described as a set of inputs, the behavior and its outputs.

- **Functionality**

The application is developed in such a way that any future enhancement can be easily implementable. The project is developed in such a way that it requires minimal maintenance. The software used are open source and easy to install. The application developed should be easy to install and use.

- **Reliability**

It is the maturity, fault tolerance and recoverability. The system is reliable for any number of user input and training dataset. The video dataset is also taken to detect the license plate and recognize the plate number.

- **Usability**

It is easy to understand, learn and operate the software system. The user can watch the vehicles and can know the license plate information.

- **Security**

It does not block the some available ports through the Windows firewall. The web camera has two paths and it can be enabled.

- **Robustness**

The application is developed in such a way that any future enhancement can be easily implementable. The project is developed in such a way that it requires minimal maintenance. The software used are open source and easy to install. The application developed should be easy to install and use.

- **Communications**

The application is developed in such a way that communication can be handled through camera. Similarly vehicle and its license plate is also detected on the screen through live web camera detection.

3.2 NON FUNCTIONAL REQUIREMENTS

Non-functional requirements determine the resources required, time interval, transaction rates, throughput and everything that deals with the performance of the system.

- **Maintainability**

It is easy to maintain the system as it does not require any special maintenance after download. Updates are required only if notified to the user about any. Easy maintenance is one among the features that makes this proposal most usable.

- **Portability**

The software must easily be transferred to another environment, including install ability. It is easily portable as it is implied on a regular computer. The user can access the computer from the place where the system was installed.

- **Performance**

Less time for detection of sign once the input is arrived. Similarly, the training time also less as we given limited epoch on training.

- **Accuracy**

The accuracy generated by our work is outperformed than any other existing models. We can recognize the emotion and eye drowsiness accurately through our proposed system.

3.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden for an government. For feasibility analysis, some understanding of the major requirements for the system is essential.

- **Technical Feasibility**

This study is carried out to check the technical feasibility, that is the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources and high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

- **Economic Feasibility**

This study is carried out to check the economic impact that the system will have on the government. The amount of fund that the government can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized camera had to be purchased.

3.4 SOFTWARE ANALYSIS

The project primarily focuses on vehicle detection. We implemented program in Python. The libraries required are to installed prior to execute the project. We installed CV2 for OpenCV, Keras, TesorFlow , numpy, json etc.

3.4.1 HARDWARE REQUIREMENTS

Processor	: Any Processor above 500 MHz.
Ram	: 4 GB
Hard Disk	: 250 GB
Input device	: Standard Keyboard and Mouse, Web Camera
Output device	: High Resolution Monitor.

3.4.2 SOFTWARE REQUIREMENTS

Operating System	: Windows 7 or higher
Programming	: Python 3.6 and related libraries

3.5 SOFTWARE DESCRIPTION

PYTHON : Python is a genuinely old language. It was designed by Guido Van Rossum in 1991 and Python Software Foundation developed it. Python is a simple language to write programs that gives users a chance to work faster and at same time to coordinate frameworks more proficiently. It has wide extent of uses from Web progression, logical and scientific processing (Orange, SymPy, NumPy) to graphical UIs (Pygame, Panda3D). The sentence structure of the python language is accurate and perfect; and code length is moderately short. It's easy to work in Python because it enables user to look after the issue as opposedto concentrating on the sentence structure.

CHAPTER 4

SYSTEM ANALYSIS

Systems analysis is a problem solving technique that decomposes a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose.

4.1 EXISTING SYSTEM

Vehicle's number/license plate recognition algorithm based on the very elementary technique of Templates matching/MATLAB. The algorithm takes an input image of the number plate (number plate should be dominant in the image) and after filtering the image, it performs region based operations. Then it tries to capture the character's regions in a processed binary image and with the aid of template matching outputs the string of number plate characters. It's a very basic approach to the problem but still produces the appropriate results. Must check out READ_ME.txt file before going to the command prompt.An autonomous information system has no meaning without any data, there is a need to reform vehicle information between reality and the information system. This can be achieved by human agents or by special intelligent equipment that will allow identification of vehicles by their registration plates in real environments.

- The basic problem associated with Indian license rate detection problem is lack of good quality images taken from low resolution camera.

- The existing system MATLAB, Open CV and TensorFlow to extract the contents from the license plate.

4.2 PROPOSED SYSTEM

Deep learning & Convolution neural network technology used to make the system which contain vehicle detection, license plate detection and license plate charcter recognition. The YOLOV3 algorithm uses the Darknet-53 network to greatly improve the network feature extraction capability and solve the problem of feature degradation caused by increasing the network depth. When the number of layers of the network increases, the features extracted by the network are more abstract and the semantic information is more abundant, this helps to more accurately predict the target category.

The given proposed work has shown us that the performance of vehicle and its plate recognition technique can be improved.
We are able to analyze it at different stages. For this purpose, we have an aim to develop a Convolutional -Neural Network (CNN) which is based on Automatic number plate recognition (ANPR).
The algorithm used will detect and recognize the vehicles using live video streams also.

- Number plate detection and recognition can be done accurately.
- Vehicle is also detected.
- By using Automatic license plate detection and recognition, government can monitor the live video streams and images, where theft vehicle, in parking management, State border control etc. can be done without any human intervention.

4.3 PERFORMANCE ANALYSIS

- Perform 416×416 size standardization processing on the vehicle training set picture, and take the picture with uniform format as the input of the network.
- Use the Darknet-53 network for image feature extraction.
- Extract the output of the 74th layer and the output of the 79th layer, and perform the residual mapping operation on the two. The result is the first feature.
- The output of the 85th layer and the output of the 61st layer are feature-spliced, and the result is mapped to the output of the 91st layer, and the result is taken as the second feature.
- The output of the 97th layer and the output of the 36th layer are feature-spliced, and the result is mapped to the output of the 103rd layer. The result is the third feature.
- Put the three features into the YOLO layer and train according to the set number of iterations to generate the vehicle detection weight model.
- Input the test set image into the trained network model. The model calls the trained parameters to detect the vehicle in the test set image and output the result.

CHAPTER 5

SYSTEM DESIGN

This chapter gives overview of architecture design, dataset for implementation, algorithm used and UML designs.

5.1 ARCHITECTURE DIAGRAM

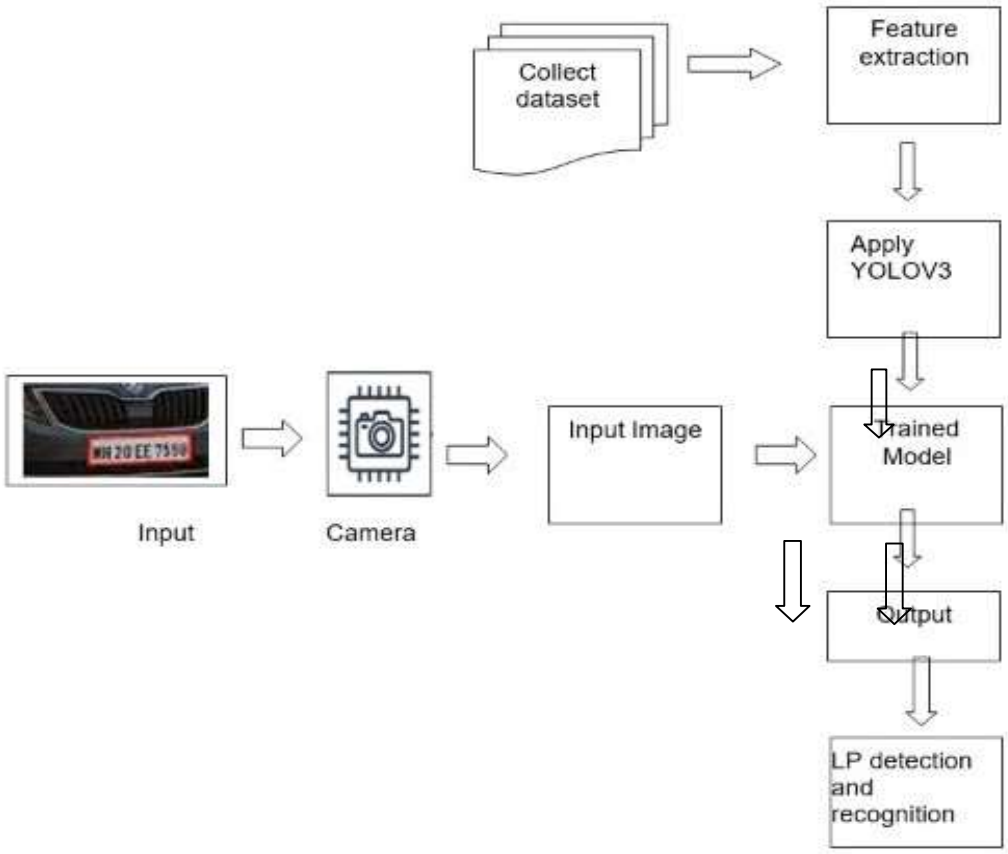

Figure 5.1 Architecture Diagram

The Figure 5.1 represents architecture of proposed system, in which all modules of the work are represented. User gives input from web camera, dataset collection, training model and recognition is mentioned.

5.2 UML DIAGRAMS

The design is a plan or drawing produced to show the look and function or workings of an object before it is made. Unified Modeling language (UML) is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. Thus, UML makes these artifacts scalable, secure and robust in execution. UML is an important aspect involved in object-oriented software development. It uses graphic notation to create visual models of software systems.
The different types of UML diagram are as follows.

- Use Case Diagram
- Activity Diagram
- Sequence Diagram
- Collaboration Diagram
- Component Diagram
- Deployment Diagram

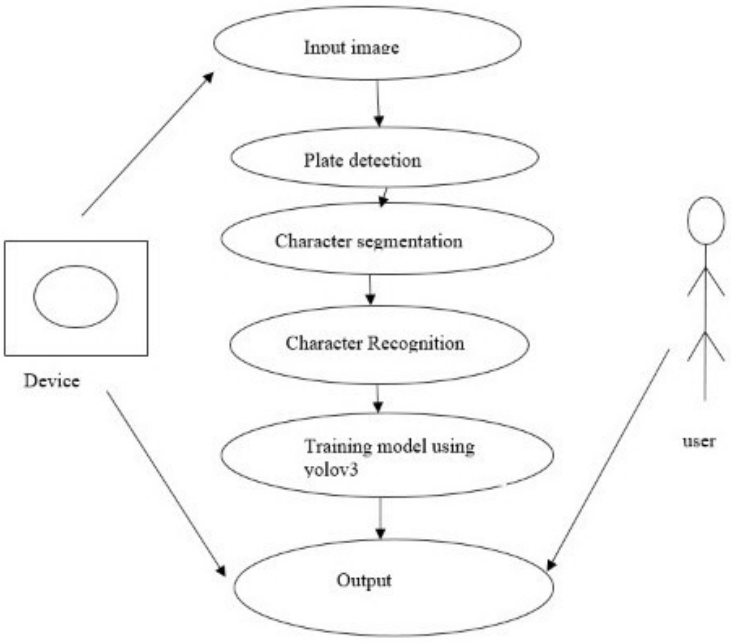

Figure 5.2 Use case Diagram

The Figure 5.2 represents use case diagram of proposed system, where user inputs dataset, the algorithm work to generate the identified output. The actor and use case is represented. An eclipse shape represents the use case namely input image, pre-process, segmentation, feature extraction, training, recognition and output.

5.3 SEQUENCE DIAGRAM

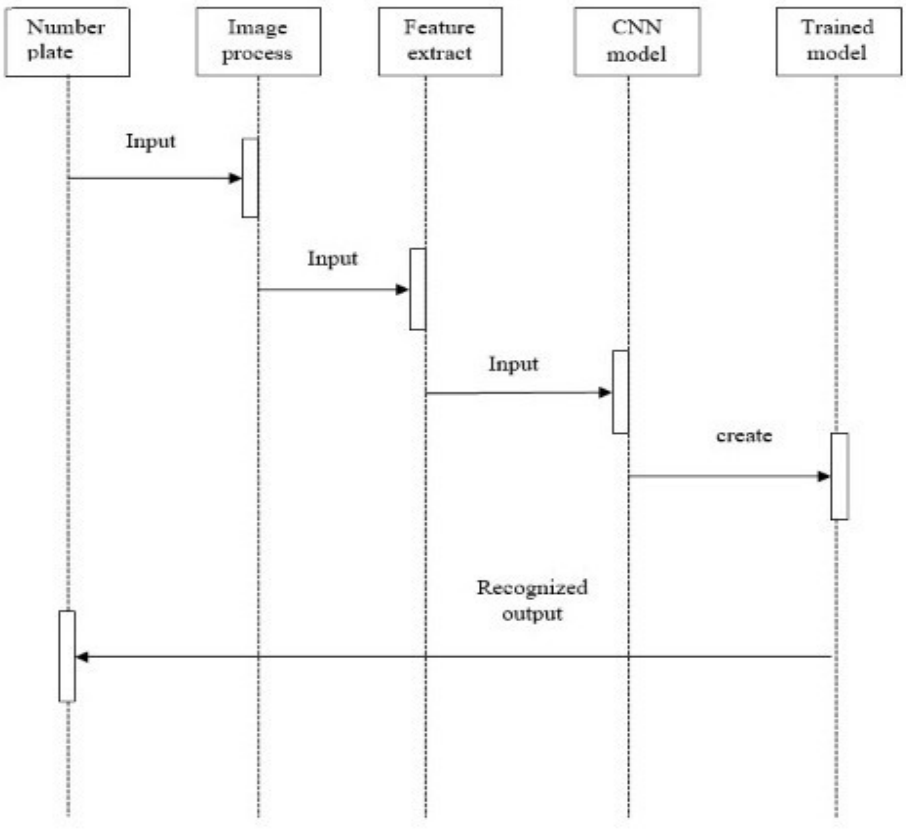

Figure 5.3 Sequence Diagram

A sequence diagram shows a parallel vertical lines, different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in order in which they occur. The above figure represents sequence diagram; the proposed system's sequence of data flow is represented in Figure 5.3.

5.4 COLLABORATION DIAGRAM

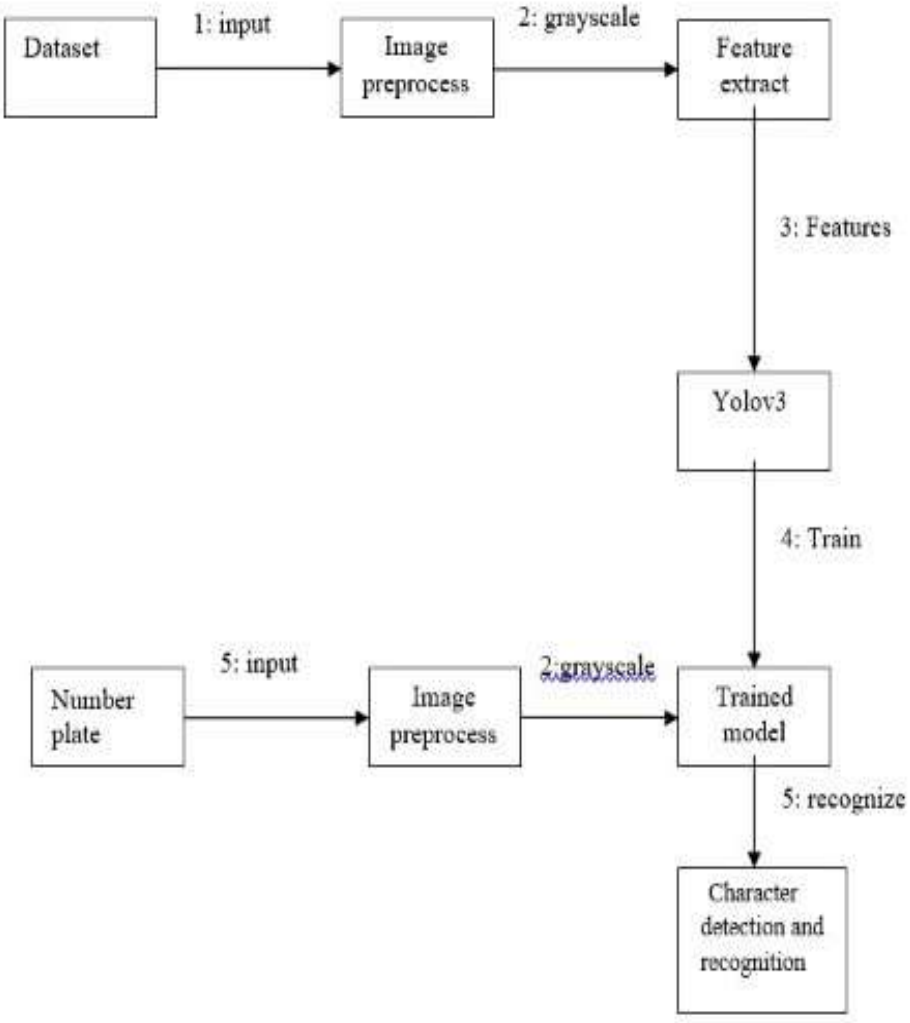

Figure 5.4 Collaboration Diagram

The Figure 5.4 shows the collaboration diagram of the proposed system, where we represented the collaboration between the actor and function modules with sequence number.

5.5 ACTIVITY DIAGRAM

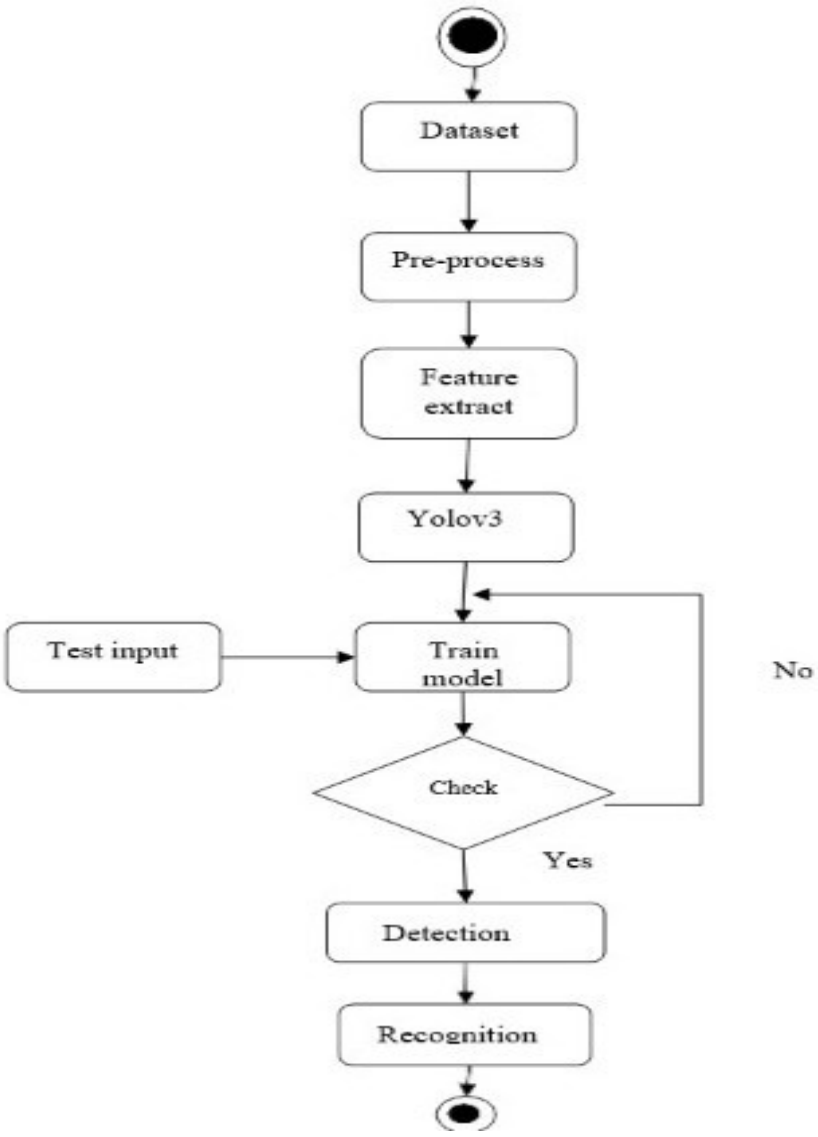

Figure 5.5 Activity Diagram

The Figure 5.5 show the activity diagram of the proposed system, where we represented the identified activities and its functional flow.

5.6 DEPLOYMENT DIAGRAM

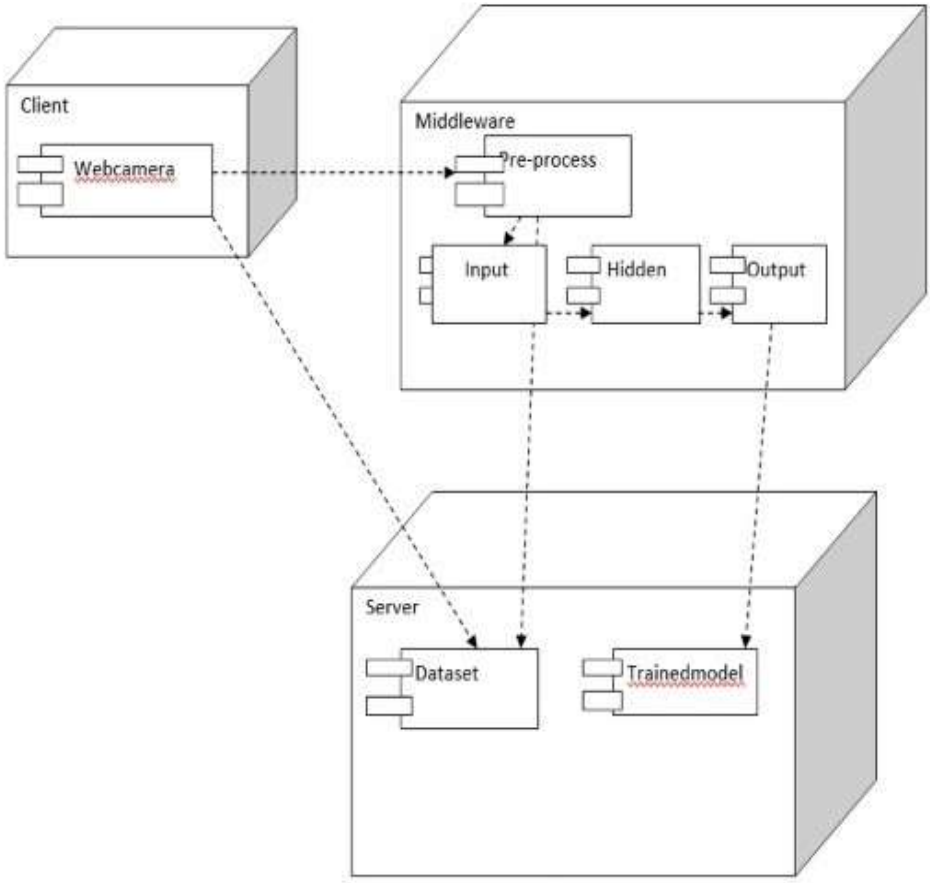

Figure 5.6 Deployment Diagram

In the deployment diagram the UML models the physical deployment of artifacts on nodes. The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster shown in Figure 5.6.

5.7 DETAILED DESIGN

- Perform 416×416 size standardization processing on the vehicle training set picture, and take the picture with uniform format as the input of the network.
- Use the Darknet-53 network for image feature extraction.
- Extract the output of the 74th layer and the output of the 79th layer, and perform the residual mapping operation on the two. The result is the first feature.

- The output of the 85th layer and the output of the 61st layer are feature-spliced, and the result is mapped to the output of the 91st layer, and the result is taken as the second feature.
- The output of the 97th layer and the output of the 36th layer are feature-spliced, and the result is mapped to the output of the 103rd layer. The result is the third feature.
- Put the three features into the YOLO layer and train according to the set number of iterations to generate the vehicle detection weight model.
- Input the test set image into the trained network model. The model calls the trained parameters to detect the vehicle in the test set image and output the result.

CHAPTER 6

METHODOLOGY

All of the methods embody the powerful idea of image features combined with machine learning technique. What they differ lie in the image feature extractors and the classifiers. Image pre-processing is carried out into steps such as color conversion. Color conversion function converts input image from one color space to other, here we used BGR2GRAY for converting the input image to gray scale image.

The facial expression dataset is the COCO dataset in the experiment. The COCO dataset has a total of 35,887 images. The dataset is divided into three parts: training set (28,709 pictures), test set (3,589 pictures), and verification set (3,589 pictures). All images are uniform 48*48 grayscale images. The dataset is divided into 7 categories, namely, anger, disgust, fear, happiness, sadness, surprise, and neutrality. Because the dataset has some noise (all black pictures, cartoon pictures, non-expression images and non-expression pictures), it is difficult to identify. The eye recognition accuracy of the COCO dataset is 65%.

When recording the UFPR-ALPR dataset, we sought to eliminate many of the constraints found in ALPR applications by using three different non-static cameras to capture 4,500 images from different types of vehicles (cars, motorcycles, buses, trucks, among others) with complex backgrounds and under different lighting conditions. The vehicles are in different positions and distances to the camera. Furthermore, in some cases, the vehicle is not fully visible on the image. To the best of our knowledge, there are no public datasets for ALPR with annotations of cars, motorcycles, LPs and characters. Therefore, we can point out two main challenges in our dataset. First, usually, car and motorcycle LPs have different aspect ratios, not allowing ALPR approaches to use this constraint to filter false positives. Also car and motorcycle LPs have different layouts and positions.

The images are pre-processed prior to training. Using data amplification techniques, 48*48pixel image blocks are taken from the upper left, lower left, upper right, lower right, and middle of each image respectively, and then flip it horizontally for a total of 10 new images. After data amplification, the numbers of the entire training set is expanded to 10 times of the original size, and then all images are subtracted from the mean value. In the network test, the test picture is input into the pretrained network model to obtain the probability of all the expression categories of the network SoftMax layer. The output of the SoftMax layer is the expression class corresponding to the maximum probability. Data amplification effectively mitigates overfitting and improves network performance. Classification is nothing but classifying the type of object like a car, bike or a person and localization is locating where the particular object is in the image by drawing lets say a bounding box over it. Object detection is a combination of both classification and localization of the object in the image. The GPU centric, efficient algorithm we are proposing to use to detect the object area from the image is the YOLO algorithm(You Only Look Once)

6.1 TRAINING USING CONVOLUTIONAL 2D NEURAL NETWORK

We used convolutional 2F neural network available in keras for traing and testing our model. The overall architecture of Conv2D is shown Figure 6.1.

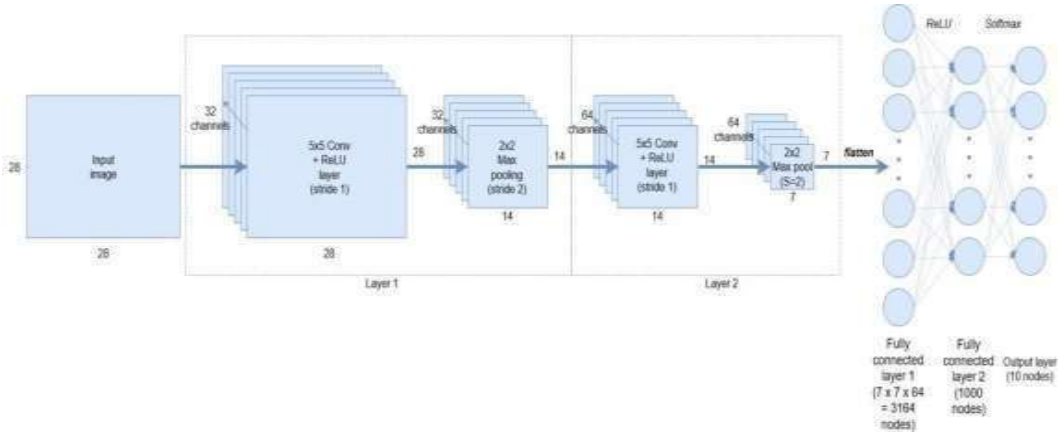

Figure 6.1: Architecture of Conv2D

There are many types of layers used to build Convolutional Neural Networks, but the ones you are most likely to encounter include:

- Convolutional (CONV)
- Activation (ACT or RELU, where we use the same of the actual activation function)
- Pooling (POOL)
- Fully-connected (FC)
- Batch normalization (BN)
- Dropout (DO)

Stacking a series of these layers in a specific manner yields a CNN. We often use simple text diagrams to describe a CNN: INPUT => CONV => RELU => FC => SOFTMAX

Here we define a simple CNN that accepts an input, applies a convolution layer, then an activation layer, then a fully-connected layer, and, finally, a softmax classifier to obtain the output classification probabilities. The SOFTMAX activation layer is often omitted from the network diagram as it is assumed it directly follows the final FC. Of these layer types, CONV and FC, (and to a lesser extent, BN) are the only layers that contain parameters that are learned during the training process. Activation and dropout

layers are not considered true “layers" themselves, but are often included in network diagrams to make the architecture explicitly clear. Pooling layers (POOL), of equal importance as CONV and FC, are also included in network diagrams as they have a substantial impact on the spatial dimensions of an image as it moves through a CNN.

CONV, POOL, RELU, and FC are the most important when defining your actual network architecture. That’s not to say that the other layers are not critical, but take a backseat to this critical set of four as they define the actual architecture itself.

6.1.1 SEQUENTIAL MODEL

Models in Keras can come in two forms – Sequential and via the Functional API. For most deep learning networks, the Sequential model is likely. It allows to easily stack sequential layers (and even recurrent layers) of the network in order from input to output. The first line declares the model type as Sequential().

6.1.2 ADDING 2D CONVOLUTIONAL LAYER

Add a 2D convolutional layer to process the 2D input images. The first argument passed to the Conv2D() layer function is the number of output channels – in this case we have 32 output channels. The next input is the kernel size, which in this case we have chosen to be a 5×5 moving window, followed by the strides in the x and y directions (1, 1).

In general Neural Network consists of different hidden layer. In most of Conv2D will have two hidden layers with 16 or 32 neurons and more, Hidden layers are multiplied with different random weight of image pixel data which is between 0 to 1. But in Conv2D Neural Network was design with same two hidden layers and each hidden layers consists of large set of neurons i.e. we used 128 dense neurons are taken and this are multiplied with random weights. By using this deep network we got promising results.

In this testing phase we used five layered Convolutional Neural Networks (CNN) model with Adam Optimization. On them one layers for convolutional, one layers for max pooling or sub sampling, one Flatten layer which converts this testing phase we used five layered Convolutional Neural Networks (CNN) model. Next, the activation function is a rectified linear unit and finally we have to supply the model with the size of the input to the layer. Declaring the input shape is only required of the first layer – Keras is good enough to work out the size of the tensors flowing through the model from there.

6.1.3 ADDING 2D MAX POOLING LAYER

Add a 2D max pooling layer. We simply specify the size of the pooling in the x and y directions – (2, 2) in this case, and the strides. The main difference between FER for still images and for video sequences is that the landmarks in the latter are tracked frame- by-frame and the system generates new dynamic features through displacement between the previous and current frames. Similar classification algorithms are then used in the video

sequences. Although this method is implemented in real time, the recognition accuracy tends to be degraded because it cannot reflect local variations of the facial components to the feature vector. Unlike a global-feature-based approach, different face regions have different levels of importance.

Apart from FER of 2D images, 3D and 4D (dynamic 3D) recordings are increasingly used in expression analysis research because of the problems presented in 2D images caused by inherent variations in pose and illumination. 3D facial expression recognition generally consists of feature extraction and classification.

One thing to note in 3D is that dynamic and static system are very different because of the nature of data. Static systems extract feature from statistical models such as deformable model, active shape model, analysis of 2D representations, and distance-based features. In contrast, dynamic systems utilize 3D image sequences for analysis of facial expressions such as 3D motion-based features. For FER, 3D images also use the similar conventional classification algorithms. Although 3D-based FER showed higher performance than 2D-based FER, 3D and 4D- based FER also has certain problems such as a high computational cost owing to a high resolution and frame rate, as well as the amount of 3D information involved.

6.1.4 ADDING ANOTHER CONVOLUTIONAL + MAX POOLINGLAYER

Next we add another convolutional + max pooling layer, with 64 output channels. The default strides argument in the Conv2D() function is (1, 1) in Keras, so we can leave it out. The default strides argument in Keras is to make it equal to the pool size.The input tensor for this layer is (batch_size, 28, 28, 32) – the 28 x 28 is the size of the image, and the 32 is the number of output channels from the previous layer.

6.1.5 FLATTEN AND ADDING DENSE LAYER

Next is to flatten the output from these to enter our fully connected layers. The next two lines declare our fully connected layers – using the Dense() layer in Keras, we specify the size – in line with our architecture, we specify 1000 nodes, each activated by a ReLU function. The second is our soft-max classification, or output layer, which is the size of the number of our classes.

6.1.6 TRAINING NEURAL NETWORK

In the training model, we have to specify the loss function, or told the framework what type of optimizer to use (i.e. gradient descent, Adam optimiser etc.).
Lass function of standard cross entropy for categorical class classification (keras.losses.categorical_crossentropy). We use the Adam optimizer (keras.optimizers.Adam). Finally, we can specify a metric that will be calculated when we

run evaluate() on the model.

We first pass in all of our training data – in this case x_train and y_train. The next argument is the batch size. In this case we are using a batch size of 32. Next we pass the number of training epochs (2 in this case). The verbose flag, set to 1 here, specifies if you want detailed information being printed in the console about the progress of the training.

6.1.7 YOLO

YOLO[15] then uses an extremely distinctive idea in object detection. Unlike the region-based methods, YOLO uniformly divides the image into cells. In each cell, there could be several object detectors, each of which has the class and the bounding box of one object in the cell. During training, the deep neural network is built, takes images from the training dataset as input, forms a loss function that incorporates the cross entropy loss, the L2 regression loss and the randomness and back propagates the gradients to update the parameters. Thus, during testing, the network output contains whether number plates exist and if yes their bounding boxes.

When YOLO came into the picture of Deep Learning Object detection algorithms, many algorithms were proposed to detect the object from the image but YOLO took a completely different approach. It was not the old classifier again produced to be an object detector. The fact that YOLO actually looked only once which the name shows and detected the object attracted many new deep learning developers. It actually looked only once but in a clever way. The first version of YOLO was proposed in 2016 and the latest version of YOLO which is YOLOv4 was proposed in 2020 which we are going to talk about in this research paper.

YOLO object detector is faster and more accurate when we compare it with other traditional object detectors, it is trained on COCO dataset for training purposes and has one stage detector which makes it faster. As we know the modern object detectors require more GPUs for training and also they train with large mini batched size, and this is what makes them slow when we actually use them to train our models. Sometimes the training time is so high that it becomes impractical to use such models. YOLOv4 proposed this issue by making object detection possible using a single GPU and smaller mini batched size. Which makes it superfast to train with a single 1080Ti or 2080Ti GPU. When the algorithm is pretty efficient but with countries like in India we should also take care that the images we are using should be better and help our algorithm to some extent to make accuracy of the overall model more. Some of the ideas which we can focus on to improve camera performance is to focus on different camera settings like the iris size, shutter speed of the camera, what is the angle of the camera and the position of the camera?. Some more things on which we can focus are the lightning conditions and type and size of lens.

YOLO first divides the image in 13*13 of which each cell detects five bounding boxes (the bounding box is the rectangle which encloses the object). YOLO gives a confidence score which shows how confident the algorithm is that the object to detect exists in the bounding box. The score doesn't tell the type of object but tells the confidence score if it is in that bounding box or not. Now we know there are n*n grid cells and each detects 5 bounding boxes, so total bounding boxes will be n*n*5 that is 185. For example there are 13*13 grid cells and each cell detects 5 bounding boxes so total bounding boxes becomes 13*13*5 which makes 845 bounding boxes. Now the beauty of the algorithm is that these total 845 boxes were predicted all at once as the name suggests “You Only Look Once”.

R-CNN takes the huge amount of time to train the model, however YOLO takes comparatively very less amount of time to train the same model.R-CNN can’t be used for real time implementation of object detection as it takes 47 seconds for each test image but YOLO can be used real time very efficiently.

6.2 ALGORITHM USED

Yolov3: Yolo abbreviates to You Only Look Once depicting its ability to detect objects and entities by using CNN (Convolutional Neural Network).Neural Network in YOLO uses weights trained by the user through annotated training data by using bounding boxes. Hence YOLO takes an image as input puts it through a Neural Network and gives the output in the image through bounding boxes.The input image is divided into S*S grid of cells. Each cell contributes to the object detection. Each cell predicts Bounding Boxes as well as Class probabilities. The prediction consists of 5 components (x,y,w,h,confidence).(x,y) represents the centre of the bounding box and (w,h) are the width and the height of the boxes Confidence represents the Estimated Prediction Accuracy of the object. YOLO is extremely fast and accurate as compared to other algorithms and hence was our primary choice for this project.

6.3 IMPLEMENTATION:

In this section, we briefly review several recent works that approaches in the context of ALPR. More specifically, we discuss works related to each ALPR stage, and specially studies works that not fit into the other subsections. This section concludes with final remarks.

6.3.1 PLATE DETECTION

Using Darkflow, we trained a YOLO (You Only Look Once) model, with 1900 images of car with annotated plate. LabelImg is a great tool witch allowed us to annotate our images in Pascal VOC format. This dataset was composed of car images we found online, some we took on the street and data augmentation (Vertical Flip, Brightness modification) using Keras (ImageDataGenerator).

6.3.2 CHARACTER SEGMENTATION

ALPR systems based on DL techniques usually address the character segmentation and recognition together. Montazzolli and Jung [20] propose a CNN to segment and recognize the characters within a cropped LP. They have segmented more than 99% of the characters correctly, outperforming the baseline by a large margin. It is very high accuracy in LP recognition jointly performing the character segmentation and recognition using Hidden Markov Models (HMMs) where the most likely LP was determined by applying the Viterbi algorithm. Character Recognition.

6.3.3 CHARACTER RECOGNITION

We train a CNN with Tensorflow and Keras libraries. There are 35 classes (10 for numbers and 25 for alphabet without “O”). We used approximately 1000 images for each classes**.** We collected a sample of characters’ images and then practiced data augmentation (rotation and brightness).

The use of random CNNs to extract features for character recognition, achieving a significantly better performance than using image pixels or learning the filters weights with back-propagation. Li and Chen [5] proposed to perform the character recognition as a sequence labelling problem. A Recurrent Neural Network (RNN) with Connectionist Temporal Classification (CTC) is employed to label the sequential data, recognizing the whole LP without the character-level segmentation.

Figure 6.2: The characters are now segmented and template matching or ANN is used to recognise the characters.

6.3.4 FINAL REMARKS:

In addition, most of the approaches are not capable of recognizing LPs in real-time, making it impossible for them to be applied in some applications. In this sense, we employ the YOLO object detection CNNs in each stage to create a robust and efficient end-to-end ALPR system. In addition, we perform data augmentation for character recognition, since this stage is the bottleneck in some ALPR systems.

CHAPTER 7

SYSTEM TESTING

After finishing the development of any computer based system the next complicated time consuming process is system testing. During the time of testing only the development company can know that, how far the user requirements have been met out, and so on.

Software testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding. The increasing feasibility of software as a system and the cost associated with the software failures are motivated forces for well-planned through testing.

7.1 TESTING OBJECTIVES

These are several rules that can save as testing objectives they are:

- Testing is a process of executing program with the intent of finding anerror.
- A good test case is one that has a high probability of finding an undiscovered error.

Testing procedures for the project is done in the following sequence

- System testing is done for checking the server name of the machines being connected between the customer and executive.
- The product information provided by the company to the executive istested against the validation with the centralized data store.
- System testing is also done for checking the executive availability toconnected to the server.
- The server name authentication is checked and availability to thecustomer
- Proper communication chat line viability is tested and made the chat system function properly.
- Mail functions are tested against the user concurrency and customer maildate validate.

Following are the some of the testing methods applied to this effective project:

The testing is a process where all the test cases are tested for the errors and they are rectified. This process is done for each of the unit and tested individually.

7.2 UNIT TESTING

Once detected, the camera was triggered to capture the image of the plate number. The system was tested on a simulated prototype road, and the calibration was to detect a speed limit of 40 centimeters/hour. Cars were made to pass on this road with varying speeds. Out of the cars that passed, the system was able to detect 3 cars moving with speeds more than the stipulated limit. The camera was placed in such a way that the rear of the car will be captured so that its plate number can be captured and verified and Table 7.1 represents a brief of test cases we are used.

Table 7.1 Test Cases

Sr. No	Test Case ID	Test Description	Test Procedure	Test Input	Expected Result	Actual Result
1	T101	To check training	Start training dataset	Execute train.py	Training should start and created output files	Trained weight file in h5 format is created
2	T102	To check image prediction	Image input	Execute predict.py	Prediction should start and output will come for image	Successfully completed
3	T104	To check recognition	Start recognition by input video or video	Execute recognize.py	Output of corresponding emotionvalue	Number plate detection with recognition

The fascinating improvement in deep learning is that it requires no need of features descriptor. Instead discriminate features are automatically extracted in this paradigm. A deep learning architecture such as CNN is composed of multilayer stacks where the convolution layers with filters are responsible for features extractor and the fully connected layer do classification. Deep CNN is being used with high accuracy in the image understanding and image classifications. First layer of the networks determines the size of input image. The input size is 30-by-30-by-1 which refers to the height, width, and the channel size of the image. The convolutional layer is tailed by a nonlinear activation function named rectified liner unit (RELU). Max-Pooling layer is used for down-sampling operation to minimize the parameters size and as another way of avoiding overfitting. It makes the features more nourished against noise and alteration. Fully linked layer combines all of the features. The softmax triggering function for classification is used by the entirely connected layer. Finally classification layer uses the probabilities returned by the softmax stimulation function to assign class into the mutually exclusive categories.

CNN trains the network after extracting features. CNNs classify images using these features. The simulation is done using MATLAB 2017a. We trained the network with 80% of dataset. The training statistics is shown in TABLE III. Maximum number of epochs was set at 15 and training started by choosing an initial learning rate of 0.0001.

The mini-batch loss is the cross entropy function for K Mutually exclusive classes and the mini-batch accuracy is the percentage of images in the current mini-batch that the network correctly classifies. The accuracy rate of the LPR using CNN is 99.6% which is remarkable.

7.3 ACCEPTANCE TESTING

Acceptance testing is a level of software testing where a system is tested for acceptability. Acceptance Testing is the fourth and last level of software testing performed after System Testing and before making the system available for actual use. The purpose of this test is to evaluate the system's compliance with the businessrequirements and assess whether it is acceptable for delivery. Firstly, the basic tests are executed, and if the test results are satisfactory then the execution of more complex scenarios are carried out. Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity todetermine whether or not to accept the system.

CHAPTER 8

RESULTS

The proposed system included two stages: the license plate detection stage and the character recognition stage. In the first stage, the license plate candidates are generated based on vertical edges to detect the potential license plate candidates. For character recognition, the comparative study was done on the AOLP dataset and also gives satisfying results. The experiment results are satisfied and encouraging and show the efficiency of the proposed system.

The following Table 8.1 shows the detection classes:

Table 8.1: Detection classes for vehicles

Class value	Detection classes of Vehicles
0	Car
1	Auto
2	Truck
3	Bus
4	Bike

To evaluate our trained network. For each of the images in our testing set, we present them to the network and ask it to predict what it thinks the label of the image is. We then tabulate the predictions of the model for an image in the testing set.

Finally, these model predictions are compared to the ground-truth labels from our testing set. The ground-truth labels represent what the image category actually is. From there, we can compute the number of predictions our classifier got correct and compute aggregate reports such as precision, recall, and f-measure, which are used to quantify the performance of our network as a whole.

SOURCE CODE

```
import cv2
import numpy as np
import time
import requests
import json

pname = 0

def get_plate_number(path):
    global pname
    regions = ['in'] #  country India
    with open(path, 'rb') as fp:
        response = requests.post(
            'https://api.platerecognizer.com/v1/plate-
reader/',data=dict(regions=regions),
# Optional
            files=dict(upload=fp),
            headers={'Authorization':'Token
b565a03a76bac29d2d04a1ea279bd6f69b006de5'})
    try:
        plate_number = response.json()['results'][0]['plate']
        pname = plate_number.upper()
        print(pname)
        sms(pname)
    except:
        pass


net = cv2.dnn.readNet("obj.weights","yolov3-tiny.cfg") # Original yolov3
#net = cv2.dnn.readNet("yolov3-tiny.weights","yolov3-tiny.cfg") #Tiny Yolo
classes = []
with open("coco.names","r") as f:
    classes = [line.strip() for line in f.readlines()]


layer_names = net.getLayerNames()
outputlayers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]


colors= np.random.uniform(0,255,size=(len(classes),3))

#loading image
cap=cv2.VideoCapture("Indian Number Plate Recognition.mp4") #0 for 1st
```

```
webcam
font = cv2.FONT_HERSHEY_PLAIN
starting_time= time.time()
frame_id = 0
obj_id = 0
calculatePeopleCount = True
peoplecount = 0
while True:
    _,frame= cap.read() #
    frame_id+=1

    height,width,channels = frame.shape
    #detecting objects
    blob                                                                    =
cv2.dnn.blobFromImage(frame,0.00392,(380,380),(0,0,0),True,crop=False)
#reduce 416 to 320


    net.setInput(blob)
    outs = net.forward(outputlayers)
    #print(outs[1])

    #Showing info on screen/ get confidence score of algorithm in detecting an
object in blob
    class_ids=[]
    confidences=[]
    boxes=[]
    TrackedIDs = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5 and class_id == 5:
                #onject detected

                center_x= int(detection[0]*width)
                center_y= int(detection[1]*height)
                w = int(detection[2]*width)
                h = int(detection[3]*height)

                #get ID
                Id = int(obj_id)
                #rectangle co-ordinaters
```

```
                x=int(center_x - w/2)
                y=int(center_y - h/2)

                #cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

                boxes.append([x,y,w,h]) #put all rectangle areas
                confidences.append(float(confidence)) #how confidence was that object
        detected and show that percentage
                class_ids.append(class_id) #name of the object tha was detected

        indexes = cv2.dnn.NMSBoxes(boxes,confidences,0.4,0.6)

        for i in range(len(boxes)):
            if i in indexes:
                x,y,w,h = boxes[i]
                label = str(classes[class_ids[i]])
                confidence= confidences[i]
                color = colors[class_ids[i]]
                cv2.circle(frame,(center_x,center_y),1,(0,255,0),2)
                #cv2.putText(frame,"peoplecount"+str(),(0,30),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,255,255), 2)
                cv2.rectangle(frame,(x,y),(x+w,y+h),color,2)
                if(frame_id % 25 == 0):
                    frame = frame[y:y+h,x:x+w]
                    cv2.imwrite("violated.jpg",frame)
                    path = "violated.jpg"
                    get_plate_number(path)

                cv2.putText(frame,label+"
        "+str(round(confidence,2)),(x,y),font,1,(255,255,255),2)
        elapsed_time = time.time() - starting_time
        fps=frame_id/elapsed_time
        cv2.putText(frame,"FPS:"+str(round(fps,2)),(10,50),font,2,(255,255,255),1)
        cv2.putText(frame,"Plate:"+str(pname),(500,50),font,2,(255,255,255),1)
        cv2.imshow("Image",frame)

        key = cv2.waitKey(1) #wait 1ms the loop will start again and we will process
        the next frame

        if key == 27: #esc key stops the process
            break;

    cap.release()
cv2.destroyAllWindows()
```

Figure 8.1 : First the image is converted into a grayscale image and Gaussian blur is applied to smooth out the image.

As shown in figure 8.1 and 8.2 we got around 91% mAP which is comparable to the human eye. The accuracy of the model for normal conditions that we were testing was 98% to localize the number plate and it was detecting the number plate with more than 90% confidence that means. We can apply a threshold of 0.9 in applications to remove unwanted plates in frame.

The speed of the algorithm and we got astonishing results too. The model is detecting the number plate in around 08 - 09seconds on images which is comparable to the latency of the human eye. While testing on videos we got around 9-13 FPS with the same mAP .

Figure 8.2 : The image is dilated to increase the boldness or brightness of the characters.

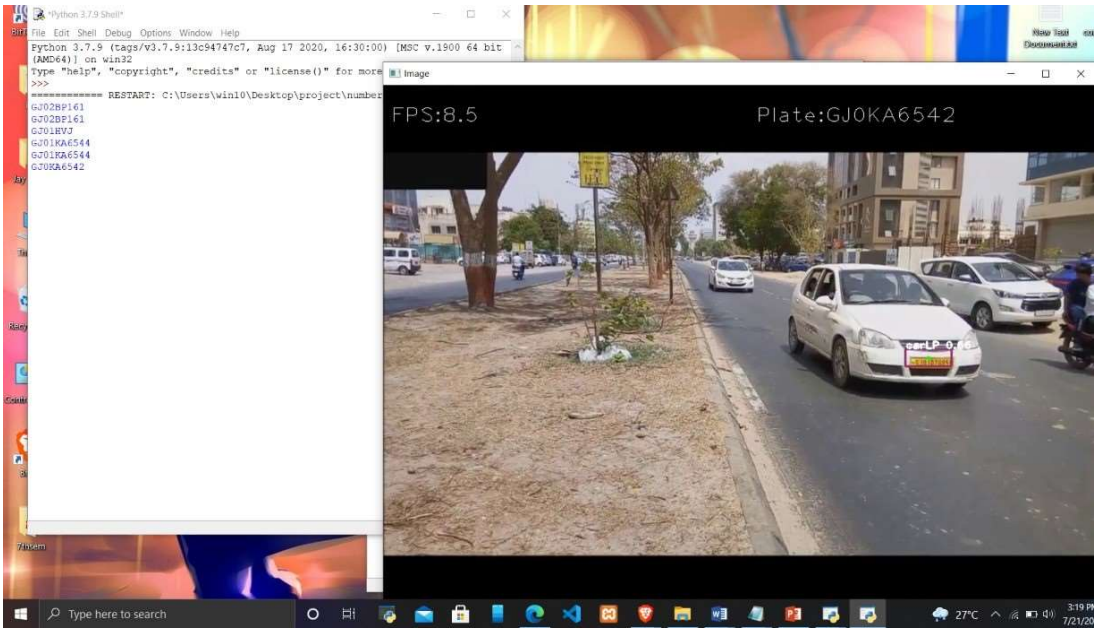

Figure 8.3 : Now the image is flipped and using OpenCV the contours are generated and characters are filtered on the basis of their height and area.

The results shown in figure 8.3 that proposed the method gives high accuracy for the three sequences and confirms that it can effectively detect the license plate and recognize its characters in different situations. Detection accuracy goes up to 98.4% and recognition accuracy goes up to 98.9% in some cases.

CHAPTER 9

CONCLUSION AND FUTURE ENCHACMENT

In this project, we have trained the robust end-to-end real-time ANPR system using the YOLO algorithm. We have also collected a mass dataset for car ANPR which includes 6,500 completely annotated real-world images. Presently, the bottleneck of our ANPR model is that it's not 100% accurate in character recognition. Although our system is able to achieve a recognition rate of 91% in character recognition, achieving 100% accuracy is very essential when implementing AN PR in commercial markets. In this sense, we have performed data augmentation technique to simulate number plates, which in turn increases the number of characters with only a few images available in the training set. Though this is a very simple technique, these essential strategies were helpful to achieve good results.

There is an immediate need of such kind of Automatic Number Plate Recognition system in India as there are problems of traffic, stealing cars etc. Government should take some interest in developing this system is very economical and eco-friendly, if applied effectively. We are able to detect and recognize the car license plate using YOLO V3. This changes will help in the progress of the nation.

APPENDIX-A

PUBLICATION DETAILS

1. JAYALAKSHMI M, HEMALATHA S, LATHIK MOGER, HEMANTH KUMAR NP "Automatic License Plate Detection And Recognition Using Deep Learning" in International Journals of Creative Research Thoughts(IJCRT), vol 9, issue 7, 20th July, 2021.

Published URL: http://www.ijcrt.org/viewfull.php?&p_id=IJCRT2107244

APPENDIX-B

PROJECT ASSOCIATES INFORMATION

	**Ms. Jayalakshmi M** pursuing Bachelor of Engineering in Computer Science and Engineering from Visvesvaraya Technological University. She is a student of Alva's Institute of Engineering and Technology, Mijar, Moodbidri. Email-ID: jayanaik9090@gmail.com Contact no: 6362314293
	**Ms. Hemalatha S** pursuing Bachelor of Engineering in Computer Science and Engineering from Visvesvaraya Technological University. She is a student of Alva's Institute of Engineering and Technology, Mijar, Moodbidri. Email-ID: hemalathassrinivas@gmail.com Contact no: 9353616601
	**Ms. Lathik Moger** pursuing Bachelor of Engineering in Computer Science and Engineering from Visvesvaraya Technological University. She is a student of Alva's Institute of Engineering and Technology, Mijar, Moodabidri. Email-ID: lathikabhaskar955@gmail.com Contact no: 6360362122
	**Mr. Hemanth Kumar N P** Senior Assisstant Professor, Department Computer Science and Engineering from Visvesvaraya Technological University. Alva's Institute of Engineering and Technology, Mijar, Moodabidri. Email-ID: hemanthnp@aiet.org.in Contact no: 9620000690

REFERENCES

[1]. Gou, K. Wang, Y. Yao, and Z. Li, “Vehicle license plate recognition based on extremal regions and restricted boltzmann machines,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1096–1107, April 2016.

[2]. R. Laroca, E. Severo, L. A. Zanlorensi et al., “A robust realtime automatic license plate recognition based on the YOLO detector,” in Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–10, Rio de Janeiro, Brazil, July 2018.

[3]. Salah Al-Shami, Ali El-Zaart, Rached Zantout, Ahmed Zekri and Khaled Almustafa. 2015 “A New Feature Extraction Method for Licence Plate Recognition”, In 2015 Fifth International Conference on Digital Information and Communication Technology and Its Applications (DICTAP), Page(s) 64–69, IEEE 2015.

[4]. Nicolas Thome, Antoine Vacavant, Lionel Robinault, and Serge Miguet, "A cognitive and video-based approach for multinational License Plate Recognition", Machine Vision and Applications, Springer-Verlag, pp. 389-407, 2011.

[5]. Hsu, G.-S.; Chen, J.-C.; Chung, Y.-Z.; , "Application-Oriented License Plate Recognition," Vehicular Technology, IEEE Transactions on , vol.62, no.2, pp.552-561, Feb. 2013.

[6]. Wu, F. Iandola, K. Keutzer, and P. H. Jin , “SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), July 2017, pp. 446–454.

[7]. Redmon, J. and Farhadi, A. (2018) YOLO v3: An Incremental Improvement.

[8]. Lin, T.-Y. et al. Microsoft COCO: Common Objects in Context. in Computer Vision – ECCV 2014 (eds. Fleet, D., Pajdla, T., Schiele, B. & Tuytelaars, T.) 740–755 (Springer International Publishing, 2014).

[9]. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 770–778 (IEEE, 2016).

[10]. Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks. IEEE Trans. Pattern Anal. Mach. Intell. 39, 1137–1149 (2017).

[11]. He, K., Gkioxari, G., Dollar, P. & Girshick, R. Mask R-CNN. in Proceedings of the IEEE international conference on computer vision 2961–2969 (2017).

[12]. Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016) You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 27-30 June 2016, 779-788.

[13]. Redmon, J. and Farhadi, A. (2017) YOLO9000: Better, Faster, Stronger. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, 21-26 July 2017, 6517-6525G.

[14]. R. Gonc¸alves, M. A. Diniz, R. Laroca, et al., “Real-time automatic license plate recognition through deep multi-task networks,” in 31th Conference on Graphics, Patterns and Images (SIBGRAPI), 110–117 (2018).

[15]. B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence 39, 2298–2304 (2017).

IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AUTOMATIC LICENSE PLATE DETECTION AND RECOGNITION USING DEEP LEARNING

[1]JAYALAKSHMI M, [2] HEMALATHA S, [3] LATHIK MOGER,

[4]HEMANTH KUMAR NP, [5]ARPITHA J C.

[1]Student, [2]Student, [3]Student, [4]Senior Assistant Professor, [4]Assistant Professor
[1,2,3,4]Department of Computer Science and Engineering
[1,2,3,4] Alva's Institute of Engineering and Technology, Mijar, India
[5]Department of Bachelor of Computer Aplications.
[5]KLE Degree College,Nagarabhavi, Bangaluru, India

Abstract: The range of vehicles has elevated dramatically over the past decades. As the number of vehicles on the road grows, it becomes more difficult for law enforcement and traffic officials to keep track of each one. Reasons of management Automatic Recognition of License Plates. ALPR (Automatic License Plate Recognition) is a common surveillance device that records vehicle license plates. It became a picture that captures and recognizes their license plate numbers. This is an essential research subject for this time period. The coco dataset was used in our research is used to recognize license plates. Continuing with the strategy of We used multitask learning to recognize character strings. YOLO-v3 changed into used for recognition, and CRNN changed into used for category and our proposed method's category. We divided the photos into three sets for evaluation: 40% for the training set, 20% for the validation set, and 40% for the test set. We chose a lower threshold for the test set evaluation, 0.125, which resulted in a 99.82 recall. Our proposed method has an 86 % recognition rate, with 88 % of three letter plates and 99 % of four letter plates being recognized. Finally, utilizing temporal redundancy, the very last identity fee is significantly enhanced, attaining 96%. Our method boosts reputation charges from 93.58 to 96.1 %, surpassing Sighthound and Open ALPR through 9% and 4.9%, respectively.

Index Terms **– CNN Model, ReadNet, YOLOv3, OpenALPR, Object Detection.**

I. INTRODUCTION

Automatic ALPR technology is gaining traction in a variety of applications, including safety and traffic control [1]. Different levels of scene analysis are required for different license plate identification applications, including identifying the categories of items in the scene, locating them, and defining the exact boundaries of each object. These scene analysis capabilities relate to three basic computer vision research tasks: image classification, object detection, and semantic (occurrence) segmentation, to name a few. Access control in structures and parking lots, law enforcement, stolen vehicle identification, traffic management, automated toll collection, and advertising learning are all common uses for License Plate Recognition Systems. There are many successful industrial systems [2] available, and there is still a lot of documentation or general public information on the ALPR system that uses deep learning algorithms for plate recognition and localization. Although, as illustrated in Figure 1, there are certain limits to cope with, such as specific detectors or viewing angles, proper illumination requirements, capturing in a predetermined region, and specific types of vehicles (they wouldn't find lps from bikes, lorries, or buses). Deep Learning (DL) methods emerged as an effective parameter in the current sector in this case. Vehicle license plate identification methods are commonly grouped into three groups based on Template Matching, characteristics, and motion information, which are frequently utilized and developed by various foreign and local researchers. Since it is clear that the demand for License Plate Detection and Recognition has existed for many years, a few research have already been conducted in this area [3][4].

The most prevalent method utilized by researchers is visual object detection, which is used as a core functional module for scene analysis in ALPR applications, piquing researchers' interest in this area. Because of the kind of open deployment environments, computerized scene evaluation at the ALPR platform will become extraordinarily demanding, posing severalth new demanding situations to item detection jobs and algorithms. The following are the principle demanding situations: (1) the way to manipulate the numerous adjustments usually encountered with inside the visible factors of items in acquiring images (for example, light, vision, small size, and ratio); (2) the way to use ALPR structures with inadequate reminiscence and computing strength at the same time as going for walks detection algorithms; (3) the way to manipulate real-time necessities and detection accuracies.

Figure 1. Lps of different layouts and notice the extensive variety in a Different format on different lp layouts.

Vehicle photos in the COCO dataset [5] can be found in a variety of locations and distances from the camera. Furthermore, the car is not often fully visible in the photograph. There aren't byany publicly to be had datasets for to the exceptional of our knowledge. ALPR with notes on cars, bikes, lps, and different objects characters as proven in FIGURE 1. As a result, we can identify two major issues. In our batch of data to begin, automotive and bicycle lps are the most common. Have various aspect ratios, preventing ALPR approaches from working. This constraint may be used to remove fake positives. Automobiles and bicycles also are available. Lps are available a number of shapes and sizes. We selected fine-tuning for ALPR due to the fact YOLO-stimulated fashions yielded sizable profits in item detection [6]. Fast-YOLO, on the other hand, is significantly faster but less exact than YOLOv3 [7], which we used in our study work. We use temporal redundancy since we're processing with video frames, so each frame were separated and then combine the findings to produce a more complete picture. For every automobile, a dependable calculation is required. According to the proposed technique is the YOLOv3 deep mastering algorithm. Outperforms COCO public automobile photos dataset consequences for a gadget that acknowledges license plates automatically.

II. OBJECT DETECTION ALGORITHM BY YOLOV3

Due to advancements in computational power (i.e., GPU and deep learning chips) and the availability of large-scale samples (e.g., COCO [8]), deep learning is becoming more popular. The learning process of a neural network is quick, scalable, and end-to-end. As a result, it has been widely utilized. The CNN model, in particular, has greatly improved picture categorization when compared to commonly used shallow approaches (e.g., Object detection, ReadNet [9]), and ReadNet [9] (e.g., Faster R-CNN [10]) and semantic segmentation (Mask R-CNN [11]), among other things. This detection framework has sparked a lot of attention, and many advanced object detectors based on it have been developed in recent years, CNN has been suggested.

Furthermore, the YOLO [12] series model is a Convolutional Neural Network-based real-time object identification system (CNN). Keeping Google's community structure in mind. The YOLO community, on the opposite hand, is doing the identical issue for cross-channel facts integration in 1x1 layers and additionally a 3x3 convolution layer. The network structure of YOLO is made up of two fully linked layers and 24 convolutional layers. Furthermore, Ali Farhadi corrected and presented YOLO v3 [7], which increases object identification performance by detecting small objects and dense or complex objects overlapping tiny items that are possibly the most common. In practical applications, a deep object detector is presented. The speed and accuracy of detection are very well balanced. Further key improvements include:

- Loss: YOLOv2's softmax loss is replaced, while the predicted object in YOLO v3 has a logistic loss. The selection of a class is more difficult when there are more variables, logistic regression is more successful. In the data collection, there are a lot of labels that overlap.
- Anchor point: YOLOv3 has nine anchor points instead of the five in YOLOv2, which improves Intersection over Union (IoU).
- Detection: YOLOv2 employs only one detecting, but YOLOv3 uses three, significantly improving the detection effects of small objects.
- Backbone: In YOLOv3, the Darknet-19 network from YOLOv2 is replaced by a darknet-53 network, enhancing object detection accuracy by extending the network. The most recent technology is used in our paper. The YOLOv3 model was used to do the detection dataset from the COCO.

A. Multitask Learning

Multi-task learning [14] is a character string recognition approach developed for licence plates. This method bypasses the character segmentation step and detects the image's character string (in this case, the cropped LP characters). Because there could be a lot of them. Each character has the ability to do a task on the network.

B. Convolutional Recurrent Neural Network

CRNN is a scene text recognition version [15] that consists of convolutional layers followed via way of means of recurrent layers, in addition to a selected transcription layer designed to create a tag collection from the per-body predictions. This layer is responsible for the input for a sequence labelling problem predicting a tag distribution x = x1, x2.... xn for each tag a single vector of features y= y1,y2…yn.

The CNN models used for license plate character detection and recognition are explained in the following sections. It's worth mentioning that all parameters (e.g., CNNS input, etc.) Are subject to change. The number of epochs, among other things, is defined here based on the validation group and as shown in the part in which the results of the experiments are reported.

III. METHODOLOGY

In this paper, we present a framework for detecting and recognizing LPs in images. Our strategy focuses on locating and reading LPs in challenging contexts. There are three essential aspects to the method:

- LPs Detection
- Character Segmentation
- Character Recognition

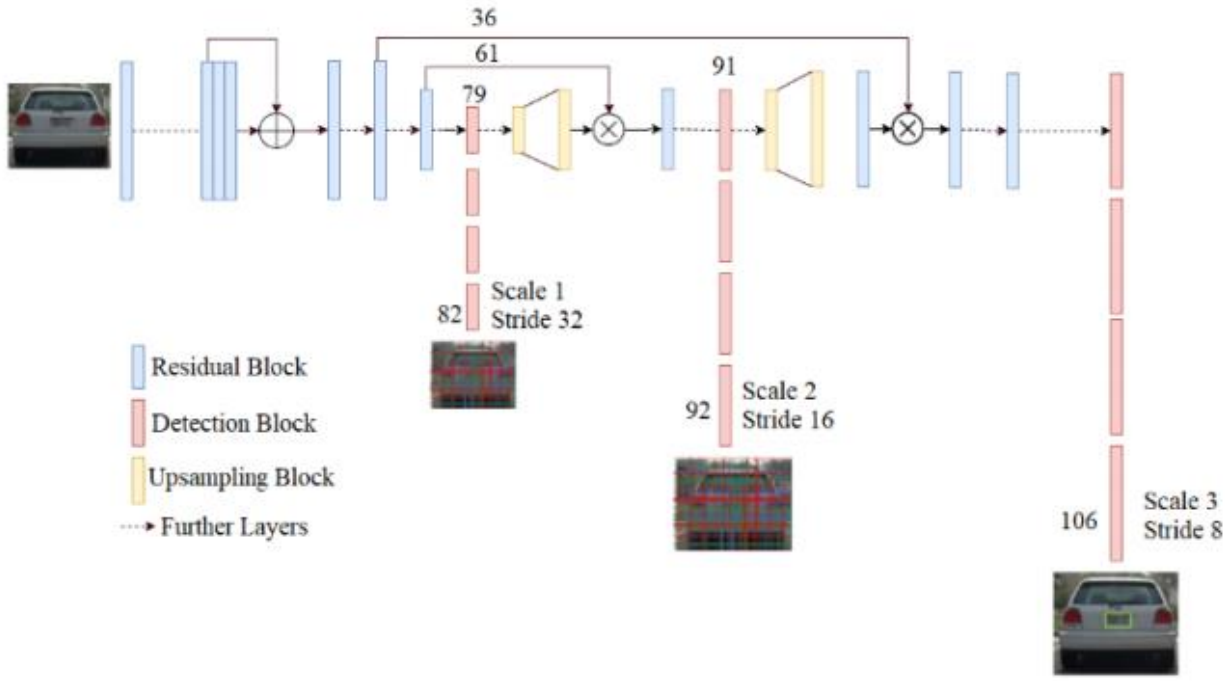

Figure 2: flow diagram of proposed alpr system

As demonstrated in FIGURE 2, we trained YOLOv3 to detect vehicles and LPs in our suggested technique. In a simple context, YOLOv3 should be able to detect LP, but in a more complex one, it may not. It might not be deep enough to perform this in complex scenarios. As a result, in order to use YOLOv3, we must make a change by considering the number of classes, the last number of layers. To forecast bounding boxes, confidence, and class probability, YOLO uses predicting boxes (A=9) and K class probability as given in Eq. (1).

$$\textbf{Filters} = (k+5)*A \quad \text{-----} \quad (1)$$

Vehicle and LPs coordinates are used to train the CNN for vehicle LPs detection in order to detect LPs. In order to detect all of the automobiles in the validation set, we assessed that the confidence level with the lowest rate of erroneous positives.

Following LP detection, the character segmentation method (as given in TABLE I.) is trained using LP and character margins and coordinates. This margin is based on the validation set to, as explained in the previous stage. Verify that all of the characters correspond to the expected LP size. We also generate a training set to expand the training set. Every LP has a negative picture.

TABLE I. CHARACTER SEGMENTATION CNN ARCHITECTURE

Layer	Filters	Size
Conv	32	3x3
Max		
Conv	64	3x3
Max		
Conv	128	3x3/2
Conv	64	1x1
Conv	128	3x3
Max		
Conv	256	3x3/2
Conv	128	1x1
Conv	256	3x3
Conv	512	3x3/2
Conv	256	1x1
Conv	512	3x3
Conv	35	1x1
Detection		

We initially provide some padding (1-2 pixels) after LP detection and segmentation to enhance prediction because certain characters may now be properly segmented. Units are used to train the networks by passing segmented text with labels as input. Features collected using CNN are converted into feature vectors and then utilised as an input for the LTSM layer (as given in TABLE II.), assisting with the sequence layer problem and forecasting a label distribution.

TABLE II. CRNN LAYERS

Layer	Input	Size
Conv	64	3x3/1
Max		2x2/2
conv	128	3x3/1
Max		2x2/2
Conv	256	3x3/1
Conv	256	3x3/1
max		2x2/2
conv	512	3x3/1
batch		
conv	512	3x3/1
batch		
max		2x2/2x1
conv	512	
Layer	Input	Hidden
LTSM	512x1x40	256

IV. EXPERIMENTAL RESULTS

For training, we use the following parameters: 50k iterations, lr=[1-2,1-3,1-4,1-5] with steps of 10k, 20k, and 25k iterations.

A. Application Orientated LP (COCO) Evaluation

We split the pictures into three sets for evaluation: 40% for the training set, 20% for the validation set, and 40% for the test set.

Figure 3. Detection results of lps detected by our method. Pointed detection of lps showing the robustness of algorithm even have different camera distance, illumination, angle, and several ambiguities.

- **Vehicle Detection**: To identify vehicles, we must first choose confidence levels. We are unable to recognise cars with 100% accuracy when confidence = 0.5 is used. In the validation dataset, all cars were correctly recognised with a confidence level of 0.25. As a result of this analysis, we set confidence for the test set at 0.125. We were able to get 100% recall with 99 % precision with this threshold, with only 5 false positives.
- **LP Detection:** Every vehicle with LPs was predicted within the bounding box in the validation set, as illustrated in FIGURE 3. As a result, vehicle patches were used to train the LP detection network. In both the validation and test sets, we achieved 100% recall and precision, which is an efficient consequence of our suggested technique.
- **Character Segmentation:** For the validation set assessment, we used the following confidence thresholds: 0.5, 0.25, and 0.125, all of which resulted in a 99.92% recall rate. So, to miss as few characters as possible, we set a lower threshold of 0.125, resulting in a 99.82 recall.
- **Character Recognition**: Padding values were introduced for character recognition 1 pixel for numbers and 2 pixels for characters in the validation set. We utilised data augmentation with a reversed character to obtain even better results. During the testing, we discovered that letter recognition may be enhanced by using augmentation and padding.

Our suggested technique achieved an 86% identification rate while ignoring temporal redundancy, identifying 88% of three letter plates and 99% of four letter plates. Taking use of temporal redundancy boosted results substantially. The final recognition rate, which was 96%, was considerably enhanced by using temporal redundancy. The recognition rate of our suggested technique increased from 93.58% to 96.1%(as given in TABLE III.). While exceeding Open ALPR and sighthound by 4.9% and 9%, respectively, the findings are demonstrated in the table below, demonstrating the suggested YOLOv3 model's high accuracy rate.

TABLE III. RECOGNITION RATES (%) WITH REDUNDANCY ACHIEVED BY PROPOSED SYSTEM INCLUDING PREVIOUS WORK ON ALOP DATASET.

ALPR	All Correct
Slighthound with redundancy	87.1
OpenALPR with redundancy	91.2
Proposed with redundancy	96.1

TABLE IV. TIME REQUIRED FOR NVIDIA 1080TI TO PROCESS THE ALGORITHM.

ALPR Stage	Time
Vehicle Detection	10.211ms
LP segmentation and Classification	2.31ms
Character Recognition	1.590ms
Total	14.111ms

Figure 4. The examples showing of accurately recognized LPs by our algorithm.

V. CONCLUSION

We suggested a real-time Automatic License Plate Recognition system employing YOLOv3 and CRNN in the proposal paper. In our modified network, the accuracy vs. speed trade-off was proved to be effective at every level. The suggested study deduces a unified technique for License Plate categorization and detection, which advances the findings utilising post processing criteria (redundancy). We overlooked mistakes in characters, those that were misclassified, and also in the amount of anticipated characters to be considered, according to the LP layout classes, which was necessary for achieving suitable results as shown in FIGURE 4. On the COCO datasets utilised in the studies, the average recognition rate is 96.1 %, surpassing Open ALPR and Sighthound by 4.9 % and 9%, respectively, demonstrating the efficacy of our proposed approach.

REFERENCES

[1]. Gou, K. Wang, Y. Yao, and Z. Li, "Vehicle license plate recognition based on extremal regions and restricted boltzmann machines," IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1096–1107, April 2016.

[2]. R. Laroca, E. Severo, L. A. Zanlorensi et al., "A robust realtime automatic license plate recognition based on the YOLO detector," in Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–10, Rio de Janeiro, Brazil, July 2018.

[3]. Salah Al-Shami, Ali El-Zaart, Rached Zantout, Ahmed Zekri and Khaled Almustafa. 2015 "A New Feature Extraction Method for Licence Plate Recognition", In 2015 Fifth International Conference on Digital Information and Communication Technology and Its Applications (DICTAP), Page(s) 64–69, IEEE 2015.

[4]. Nicolas Thome, Antoine Vacavant, Lionel Robinault, and Serge Miguet, "A cognitive and video-based approach for multinational License Plate Recognition", Machine Vision and Applications, Springer-Verlag, pp. 389-407, 2011.

[5]. Hsu, G.-S.; Chen, J.-C.; Chung, Y.-Z.; , "Application-Oriented License Plate Recognition," Vehicular Technology, IEEE Transactions on , vol.62, no.2, pp.552-561, Feb. 2013

[6]. Wu, F. Iandola, K. Keutzer, and P. H. Jin , "SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), July 2017, pp. 446–454

[7]. Redmon, J. and Farhadi, A. (2018) YOLO v3: An Incremental Improvement.

[8]. Lin, T.-Y. et al. Microsoft COCO: Common Objects in Context. in Computer Vision – ECCV 2014 (eds. Fleet, D., Pajdla, T., Schiele, B. & Tuytelaars, T.) 740–755 (Springer International Publishing, 2014).

[9]. He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 770–778 (IEEE, 2016).

[10]. Ren, S., He, K., Girshick, R. & Sun, J. Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks. IEEE Trans. Pattern Anal. Mach. Intell. 39, 1137–1149 (2017).

[11]. He, K., Gkioxari, G., Dollar, P. & Girshick, R. Mask R-CNN. in Proceedings of the IEEE international conference on computer vision 2961–2969 (2017).

[12]. Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016) You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 27-30 June 2016, 779-788.

[13]. Redmon, J. and Farhadi, A. (2017) YOLO9000: Better, Faster, Stronger. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, 21-26 July 2017, 6517-6525G.

[14]. R. Gonc¸alves, M. A. Diniz, R. Laroca, et al., "Real-time automatic license plate recognition through deep multi-task networks," in 31th Conference on Graphics, Patterns and Images (SIBGRAPI), 110–117 (2018).

[15]. B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence 39, 2298–2304 (2017).