# INTELLIGENT SPEED LIMIT RECOGNITION AND ENGINE CONTROL SYSTEM FOR ENHANCED ROAD SAFETY

**A PROJECT REPORT**

*Submitted by*

**AMEER ARAFATH . B**

**ARUL KARTHI . K**

**HARI PRAKASH . M**

*In partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

*in*

ELECTRONICS AND COMMUNICATION ENGINEERING

**SSM INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**DINDIGUL-624002**

**ANNA UNIVERSITY::CHENNAI-600025**

MAY 2023

# BONAFIDE CERTIFICATE

Certified that this project report "**Intelligent Speed Limit Recognition and Engine Control System for Enhanced Road Safety"** is the bonafide work of "**AMEER ARAFATH . B (922119106005), ARUL KARTHI . K (922119106009), HARI PRAKASH . M (922119106031),"** who carried out the project work under my supervision.


**SIGNATURE OF HOD**

**Dr. S. KARTHIGAI LAKSHMI,**

**HEAD OF THE DEPARTMENT**

Department of Electronics and Communication Engineering,

SSM Institute of Engineering and Technology,

Dindigul - 624002.

**SUPERVISOR**

**Mr. A. RAMACHANDRAN**

**ASSISTANT PROFESSOR**

Department of Electronics and Communication Engineering,

SSM Institute of Engineering and Technology,

Dindigul - 624002.


Submitted for the VIVA-VOCE Examination held on _____

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

First and foremost, we would like to express our deep sense of gratitude to our most honorable Chairman and Management Trustee **Mr.K.Shanmugavel** of SSM Institute of Engineering and Technology for providing us with necessary facilities during the course of study.

We feel immensely pleased to express our sincere thanks to our Principal **Dr.D.Senthil Kumaran** for encouragement and support.

We extend our solemn gratitude to **Dr.S.Karthigai Lakshmi**, Head of the Department, Electronics and Communication Engineering, for her timely support to all our activities. We express our sincere thanks to our guide **Mr.A.Ramachandran,** AP/ECE and our coordinator **Dr.K.Vinoth Kumar,** Professor/ECE for his guidance throughout our project work.

Also, we express our thanks to the faculty members of our department, nonteaching staff members and my dear friends for their moral support, help and encouragement towards the successful completion of the project. We are most indebted to our parents, with whose support, resulted in making our dreams of becoming successful graduates, a reality.

# ABSTRACT

Road safety is a critical issue, and speeding has become a leading cause of accidents in recent years. To combat this problem, a proposed system that combines a Python-based speed limit detection system and an Arduino-based motor speed regulation system offers a potential solution. The proposed system aims to regulate the speed of a motor based on the speed limit of the area in which it is being used, using machine learning algorithms and microcontrollers. The speed limit detection system is built on Python, a popular and versatile programming language. The system uses machine learning algorithms to detect and classify speed limit signs, making it possible to regulate the speed of a motor based on the speed limit of the area. The system can detect and classify different types of speed limit signs, including those indicating changes in speed limit, such as variable speed limit signs. This ensures that the motor's speed is adjusted according to the latest speed limit information available, providing an efficient solution to the problem of speeding. The motor speed regulation system uses an Arduino microcontroller to regulate the motor's speed based on the speed limit. The system can be easily implemented on different types of motors and is adaptable to various settings, including personal vehicles, public transportation, and industrial machinery. The proposed system's versatility also makes it extendable to include other features, such as automatic braking systems and collision detection systems. These additional features could further improve road safety by providing an automatic response to potential collisions. Overall, the proposed system has the potential to significantly improve road safety and reduce the number of accidents caused by speeding. Its cost-effectiveness, versatility, and scalability make it a viable solution for regulating speed limits in various settings. By combining machine learning algorithms and microcontrollers, the system provides an efficient and effective solution to the problem of speeding, offering a promising prospect for safer roads.

## TABLE OF CONTENTS

## LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

Cars became a primary means of transport in the 20th century and their number has been dynamically growing since the time they were invented. At the present the motorways and the urban roads in many developed and developing countries are full of vehicles, which has exposed the drivers to various risks. This, together with the technological progress of the post-war decades, boosted the development of car industry and the research aimed at increasing driving safety and automation of the vehicle navigation process.

First recognized computer vision-based driver assistance systems were developed in 1980's when adequate computing hardware and cameras could already be fitted in a standard passenger car. As road signs are an important part of the traffic infrastructure which plays a key role in regulating flow of the vehicles, it was soon found necessary to include in the Driver Support Systems (DSS), as they were often called, the visual traffic sign recognition (TSR) functionality. Nowadays, TSR is considered only a single aspect of the computer-aided driver assistance, along with obstacle detection, pedestrian detection, parking assistance or lane departure alerting, as well as a range of non-visual components like GPS-based vehicle positioning or intelligent route planning. There are three potential usage scenarios of a traffic sign recognition system, ordered by the level of interference of TSR in the human activity:

I. The present-day scenario: TSR is used to detect and recognize the traffic signs passed by and to present the appropriate information to the driver. No further actions are taken. This information can have a form of an arbitrary sound, an understandable natural-language audio message, or a visual presentation of the detected and interpreted sign's prototype on the dashboard, as well as their combination

II. The near-future scenario: TSR system not only detects and recognizes road signs, but is also allowed to trigger certain, limited mechanical actions preventing the vehicle from achieving various dangerous states, and hence ensuring safety of a human. Such a limited intervention could involve for example an electronic speed reduction or a fuel inflow cut-off in presence of speed limit signs, or to give the right of way to the other, privileged traffic participants.

III. The future scenario: TSR is used to detect and recognize traffic signs and is fully integrated with the other DSS components, which jointly take full responsibility for a vehicle navigation. In this scenario the vehicle would be entirely autonomous and the role of a human would be reduced to merely providing the journey endpoints and possibly some additional requirements to the system, just in a way the GPS is nowadays used.

Undoubtedly, the first stage of the TSR systems development has already been reached. However, the current sign detection and recognition algorithms are still far from perfect. First of all, the existing solutions do not seem to have achieved maturity in providing their basic functionality on a practically acceptable level. For example, the current TSR systems cannot handle all types of traffic signs. They can merely recognize narrow subcategories exhibiting similar shape and color characteristics, where the intra-category appearance variability is relatively low, e.g, speed limit signs. Recognition of more complex signs, like those giving directions, which have unstandarised shapes and colors and contain textual information to be interpreted, is still a great challenge. Secondly, there are numerous undesirable factors and driving circumstances in which the reliability of the existing TSR approaches significantly drops. These include adverse illumination, rainfall/snowfall, or vibrations which might be caused by the vehicle moving on an uneven road surface. Only when the abovementioned limitations are properly addressed, a way towards fully autonomous vehicle navigation will be open.

## 1.1 SPEED LIMIT AND RULES :

Speed limit boards are traffic signs that indicate the maximum speed limit allowed on a particular road or section of a road. The rules associated with speed limit boards vary depending on the country and jurisdiction, but generally, drivers are expected to follow these rules:

I. Obey the posted speed limit: The speed limit board indicates the maximum speed that you are allowed to drive on a particular road. Drivers should always follow the posted speed limit to avoid being ticketed for speeding.

II. Adjust your speed to the conditions: While the posted speed limit is the maximum allowed speed, drivers should adjust their speed to the current driving conditions. For example, if the road is wet or there is heavy traffic, drivers should slow down even if the posted speed limit is higher.

III. Slow down in school zones: In many countries, speed limit boards are used in school zones to indicate a lower speed limit during school hours. Drivers should always slow down in school zones to protect children and obey the posted speed limit.

IV. Watch for changes in speed limit: Speed limit boards can change throughout a road or highway, so drivers should be aware of any changes in the posted speed limit and adjust their speed accordingly.

V. Pay attention to road signs and signals: Drivers should always pay attention to road signs and signals, including speed limit boards, to ensure they are driving safely and following the rules of the road.

In general, it's important to remember that speed limit boards are there to promote safety and reduce the risk of accidents. By following the rules associated with speed limit boards, drivers can help to keep themselves and others safe on the road.

## 1.2 DEEP LEARNING :

Deep learning is a subset of machine learning that is based on artificial neural networks, which are inspired by the structure and function of the human brain. The goal of deep learning is to enable computers to learn from large amounts of data, by discovering complex patterns and representations that can be used to make predictions, decisions or classifications. Unlike traditional machine learning methods, which require expert feature engineering, deep learning algorithms learn from the raw data by automatically identifying the features that are most relevant to the task at hand. This makes deep learning particularly effective in solving problems that are difficult to express in terms of simple rules or logical expressions, such as image recognition, natural language processing, speech recognition, and autonomous driving.

One of the key features of deep learning is the use of multiple layers of artificial neurons, also known as deep neural networks. Each layer learns to represent the data at a different level of abstraction, by combining the outputs of the previous layer in a non-linear way. This hierarchical structure allows deep learning models to capture highly complex and abstract relationships between the input and output variables, which is critical for achieving state-of-the-art performance in many applications. However, training deep neural networks can be computationally intensive and requires large amounts of data, which has led to the development of specialized hardware and software frameworks for deep learning.

Despite the challenges associated with training and deploying deep learning models, their performance in many real-world applications has been truly remarkable. For instance, deep learning models have achieved human-level accuracy in tasks such as image classification and speech recognition, and have enabled significant improvements in medical diagnosis, drug discovery, and autonomous driving, among others. As a result, deep learning has become a key driver of innovation in artificial intelligence, with many researchers and companies working to improve the technology and develop new applications.

## 1.3 DIP USING PYTHON :

Image processing is a vast and complex field that involves manipulating and analyzing digital images. Python has become a popular language for image processing due to its ease of use, vast collection of libraries and tools, and strong support for scientific computing. Python provides a wide range of libraries for image processing, such as Pillow, OpenCV, Scikit-Image, and Pygame. These libraries offer various functionalities for image processing such as reading, writing, resizing, filtering, segmentation, object detection, and recognition.

One of the most popular libraries for image processing in Python is Pillow. It provides a wide range of tools for image manipulation, including image reading, resizing, cropping, filtering, and color conversion. With Pillow, images can be loaded from different file formats, such as JPEG, PNG, BMP, and GIF, and then manipulated using a wide range of built-in functions. Additionally, OpenCV is another widely-used library for computer vision and image processing. It offers various functions for image manipulation, object detection, and feature extraction. OpenCV also provides tools for face detection, motion detection, and optical character recognition, making it a powerful tool for developing complex computer vision applications.

Another popular library for image processing in Python is Scikit-Image. It is an open-source image processing library that provides a wide range of algorithms for image segmentation, filtering, and analysis. Scikit-Image includes tools for image registration, edge detection, texture analysis, and color quantization, among others. Additionally, Pygame is a popular library for creating games, simulations, and animations in Python, but it can also be used for simple image processing tasks. Pygame provides functions for loading and displaying images, as well as for creating simple animations and graphical user interfaces.

In conclusion, Python provides a powerful and flexible platform for image processing, with a wide range of libraries and tools that can be used to tackle a broad spectrum of challenges. The

ease of use, vast collection of libraries, and strong support for scientific computing make Python an ideal language for developing image processing applications. With its growing popularity and active community of developers, Python is poised to remain a dominant force in the field of image processing in the years to come.

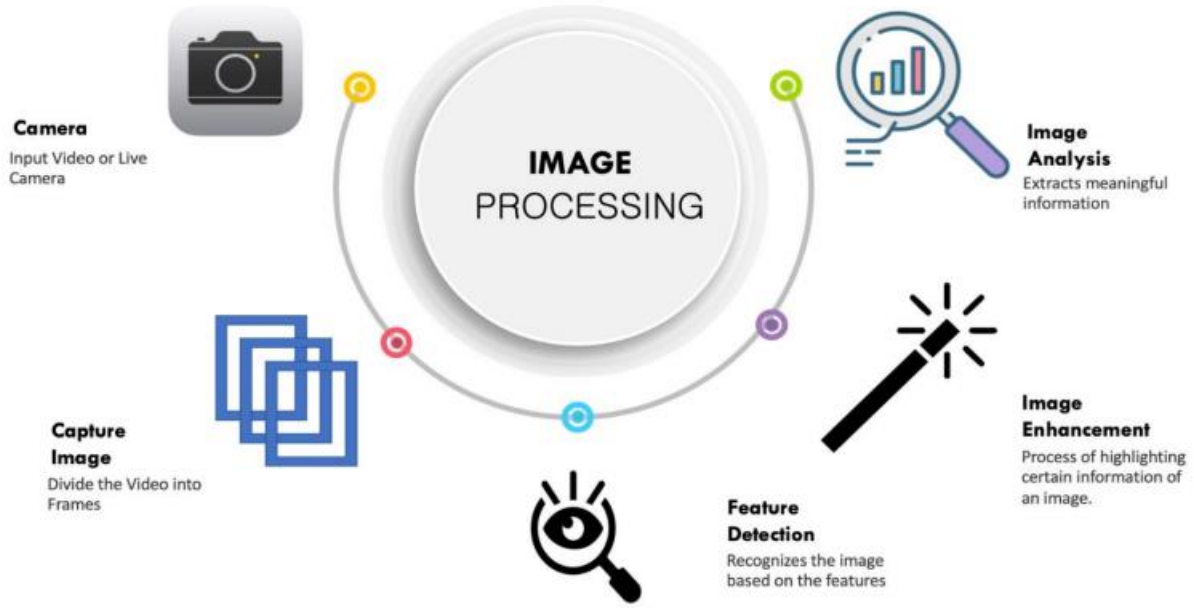



**Fig:1.1 Digital Image Processing**

## 1.4 PROBLEM FORMULATION AND GOALS

The aim of the traffic sign recognition system operating on board of a vehicle is to detect and track the sign instances over time and to correctly interpret their pictograms, so that the driver can react properly to the encountered traffic situation. The input to a TSR system is a live video stream captured by an in-vehicle camera/cameras and its output are the desired-form signals providing a human understandable interpretation of the detected and recognised signs. Such a system can be conceptually visualised using a block diagram with three main components, as shown in Figure 1.1. The arrows between each pair of components are drawn by default in both directions. However, depending on the actual system architecture, certain interactions may be unidirectional, or may not exist at all. This diagram will be presented in more detail in the following sections.

Consider a single front-looking camera mounted on board of a vehicle in front of the windscreen. Whenever a relevant traffic sign is detected within the field of view of the camera, as shown in Figure 1.2a, the TSR system should analyse the sign's pictogram over time, classify it, possibly before the sign is passed by, and present the outcome to the driver, for instance in a way shown in Figure 1.2b.

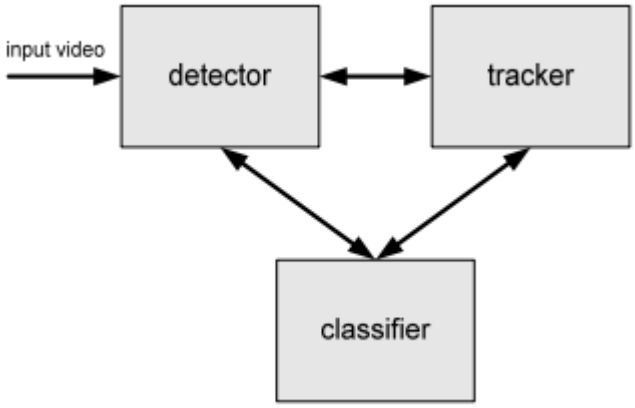



**Fig: 1.2 Block diagram of traffic sign recognition system operating on board of a vehicle.**

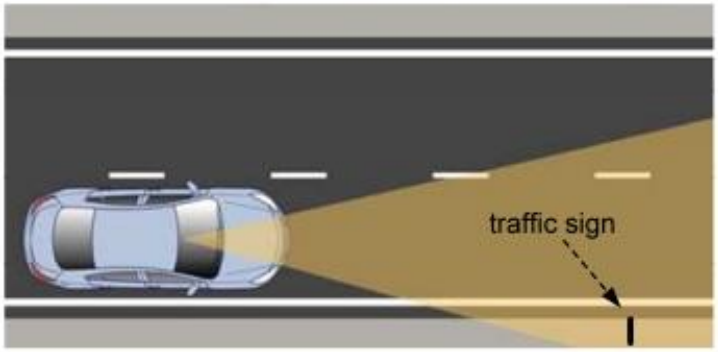



**Fig:1.3 (a)** **Fig:1.3 (b)**

Figure 1.3: Usage scenario of a traffic sign recognition system: a) schematic depiction of a vehicle approaching a traffic sign, b) an example way of presenting the information about a detected and recognised sign to the driver. The right illustration by courtesy of Siemens AG.

In the detection stage we want to capture possibly all instances of traffic signs which fall into the field of view of the camera, and which apparent size in the image is in between a predefined minimum and maximum. This apparent scale range corresponds to the range of distances of a sign from the camera and is defined such that only the signs mounted in the close vicinity of the road are included in the analysis. As scene segmentation and object detection always precede the higher-level analysis in a processing pipeline, we would like our sign detector to produce very few false alarms, even at the cost of missing certain true positives in a single image. This choice is based on an assumption that the potential cost of false positives for the driver is generally higher than the negative impact of not detecting the true signs, at least when the TSR system is only intended to play a supportive role. Besides, when video stream is processed, missing the target in one frame can always be made up for in another, where it is possibly easier to discriminate it from the background.

The goal of the road sign tracker is to maintain the track of an initially detected sign over time, until it disappears from the field of view of the camera. We want our tracker to go beyond the commonly adopted scheme in which it is only used to reduce the local search region for the

sign detector. First, we would like to model the evolution of a tracked sign, or at least its boundary, on a feature or pixel level. Secondly, our goal is to develop a framework for modelling the full structure of the affine apparent transformations of the target in the image plane so that its frontal view can always be retrieved, regardless of the actual camera's viewpoint. Secondly, the desired tracker should be able to operate efficiently in real time, in natural, possibly cluttered urban street scenes. To certain extent it should also be robust to illumination changes and ought to have an ability to retrieve the geometry of the target even when for several frames it remains occluded.

Humans are faced with the problem of recognition in their everyday lives. Whether it concerns as intangible matter as disease diagnosis, or more perceivable one, like face identification, there seems to be no universal recipe for an always working solution. In other words, every recognition problem is different. Regardless of the actual entity to be recognised, a natural approach involves identifying some number of features that make the object of interest distinguishable. It is usually possible to determine such distinctive features straightaway, based on the domain-level knowledge. However, sometimes the feature space is too large or the discriminative patterns in the data are not easy to spot. This requires aid of the advanced data analysis techniques from an area of data mining, statistical pattern recognition, machine learning, artificial intelligence and the related fields.The framework for traffic sign detection, tracking and recognition is illustrated in Figure 1.3. The system is composed of three components: detector, tracker and classifier. Unless otherwise stated, the input to our system are image sequences captured by a single car-mounted wide-angle camera.

However, the system remains fully operational if static images are passed on input. In that case the tracking component is disabled and the processing only involves detection and recognition. When performing a dynamic road sign recognition, the input video stream is assumed to be already split into individual frames. An in-vehicle TSR system should capture traffic signs at a considerable distance from the camera in order to allow time for an appropriate analysis, temporal information fusion, and ultimately a reliable recognition. As such distant signs appear nearly stationary, i.e. their apparent traffic sign candidates in every frame of the input video is unnecessary. Therefore, at the beginning of the processing pipeline the frame number is

checked. Full scene exploration is done only every n frames. In all other frames only the states of the already established sign ca ndidates are updated. Detection of new likely traffic signs is preceded by an appropriate preprocessing of the input image. It is intended to filter the image in order to extract the desired lowlevel features, e.g. compute image gradients, determine edges, or amplify patches of certain colour. The details of the image preprocessing techniques used in our approach will be discussed in Chapter 3. With a low-level feature map available, a fast, top-down, quad-tree scene analysis is performed. The goal in this step is to rapidly establish the initial regions of interest (RoI-s) where the likelihood of signs' presence is high. In principle, these regions are determined in a very conservative way, i.e. such that the risk of not capturing all true sign instances in the scene is kept to minimum, even at the cost of some RoI-s covering unnecessarily large fragments of the scene.

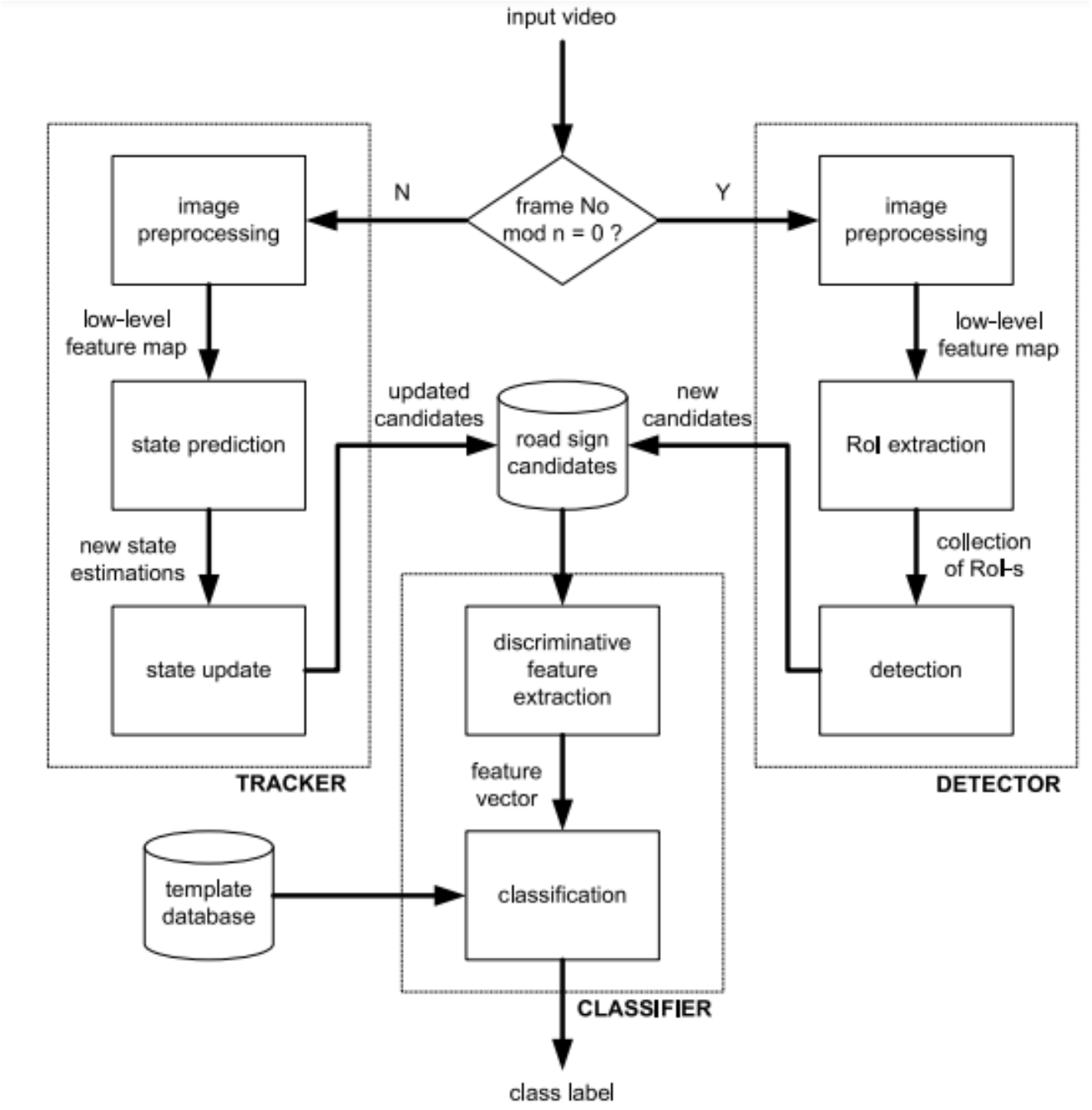



**Fig:1.4 Architecture of the traffic sign detection, tracking and recognition system**

The recognition point of view. In the recognition stage the classification component analyses each road sign candidate from the pool of tracked candidates. At this point a candidate is assumed to be a (possibly warped) centred image of a sign cropped from the current frame image. This image is first re-scaled to a predefined size. Then, depending on the actual recognition strategy adopted, it is subject to further preprocessing which facilitates extraction of the most discriminative image features that have been determined in the classifier training process. We have developed three different classifiers that are suitable for solving problems involving multiple similar object classes. However, all three classifiers are based on a common idea of representing each sign with a collection of distances/similarities to the class prototypes.

The first classifier is based on a discriminative representation of traffic signs, learned independently for each class based solely on the idealised template sign images. This representation is assembled from a compact set of local image regions where a given template differs possibly the most from all other templates with respect to a novel distance metric based on the so-called Colour Distance Transform. The two other classifiers require a number of realistic sign images for training. A soft sign similarity measure is inferred from the pairs of training images, where each pair is labelled: “same” or “different”, depending on whether or not the paired images represent the same or different pictograms.

## 1.5 Traffic Sign Recognition

Road signs are an inherent part of the traffic infrastructure. They are designed to regulate flow of the vehicles, give specific information to the traffic participants, or warn against unexpected road circumstances. Road signs are always mounted in places that are easy to spot by the drivers without distracting them from manoeuvring the vehicle, e.g. on posts by the roadside or over the motorway lanes. Besides, their pictograms are designed in a way that admits easy discrimination between multiple signs, even from the considerable distance and

under poor lightning and weather conditions. These properties of traffic signs have not changed for decades, with the exception of better, more endurable and more reflective materials being used in the production process. In addition, new sign types are constantly introduced to reflect an inevitable technological advance in the traffic infrastructure and road safety standards.

An interest in automatic road sign recognition arose naturally with the introduction of the machine vision systems and can be traced back to the research on vision-based mobile robots in the late 1960's and early 1970's. However, at that time the computational power available on a mobile platform was very limited and due to this technological gap only very few research groups ventured to work in this area. More attention to the problem was given only later, in the beginning of the digitisation era, when video processing became more attainable on a computer machine and in the same time the demand for driver comfort and safety increased. At that time the traffic sign recognition problem was considered in a much broader context of multi-aspect driver support, which set foundations for the term Driver Support Systems (DSS). This has not changed until now. Such systems are generally referred to as hardware and software tools providing continuous assistance in the routine driver's activities, giving information about the upcoming major navigation decisions and potential dangers, as well as monitoring the vehicle's state and the level of driver's safety.

Significant advances in the area of DSS were made in late 1980's and 1990's when numerous large-scale projects were developed in the USA, e.g. DARPA ALV programme 1 , IVHS project 2 , Europe, e.g. PROMETHEUS project 3 , and Japan. The main contribution of these projects was popularization of the intelligent vehicle concept, which gave birth to the numerous academic and industrial automotive research groups. It would have probably never happened if it had not been for the technological progress of the 1990's which made it possible to operate a visual driver assistance system on board of a vehicle with the use of the off-the-shelf video cameras and cheap mobile computers. In parallel, numerous research papers were published and many international discussions on DSS were initiated. One of such discussions in

Summer 1990 in Tokyo not only resulted in a valuable book by Masaki et al. [27] on vision-based vehicle guidance, but also started a series of annual Intelligent Vehicles Symposia' sponsored by the IEEE Industrial Electronics Society.

At present, visual driver assistance is one of the hot topics in the machine vision and the intelligent vehicles communities. Many groups around the world conduct research in this area and the most innovative solutions are given opportunity to be tested in the realistic traffic situations in the famous DARPA Grand Challenge [160]. Large progress has also been made recently in traffic sign detection and recognition which resulted in the first successful, large-scale commercial applications, e.g [161, 162]. Although the existing solutions expose numerous limitations, the future of this technology is bright. With the current pace of development kept, in the coming years we are likely to see millions of new cars with the automatic visual traffic sign recognition systems operating on board. Let us below briefly review the published research that led to the success of this technology we are observing today.

First published studies related to the problem of traffic sign recognition (TSR) are dated mid 1980's. They discussed different computer vision methods for sign detection and recognition involving colour segmentation, edge analysis, various forms of template matching, classification based on the neural networks and many other. An excellent compilation of these studies can be found in [39]. Since traffic signs have a priori known shapes and colours, in a vast majority of the early approaches to the TSR problem both pieces of information were exploited. The typical setup involved three-stage processing: detection, analysis and classification. In this model a global scanning of the scene was first performed in order to identify a small number of subregions which should be tested for compatibility with the hypothesis of representing a traffic sign. This preliminary detection was usually driven by colour cue analysis. Traditionally, in the second step the extracted interest regions were more carefully analysed in order to be mapped into the appropriate semantic class categories or discarded. The more accurate classification of the positively verified hypotheses was done in the final step.

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 Improved detection method for traffic signs in real scenes applied in intelligent and connected vehicles

Du et al presented a modified model based on the method of You Only Look Once (YOLO) for detecting different types of Chinese signs, including mandatory, prohibitory, danger warning, guide, and tourist signs. Images of Chinese traffic signs are collected in real scenes and a new dataset is established. The modified model combines the DenseNet method with the YOLOv3 network. Dense blocks are used to strengthen feature propagation and promote feature reuse in those feature layers with low resolution in the YOLOv3 network. Experimental results on the test dataset reveal that the average precision of the modified model, the original YOLOv3, and the YOLOv2 networks are 95.92, 94.59, and 89.39%, respectively.

## 2.2 Recognition of Vehicles, Pedestrians and Traffic Signs Using Convolutional Neural Networks

ÖZTÜRK et al described as one of the functions of an autonomous car, the operation of peripheral object recognition is carried out through the use of deep learning which has been mentioned with great accuracy and speed these years in the field of solving problems in machine learning. Signs and objects in various environments, different viewing angles and dimensions can be recognized through the video images taken from the vehicle. Application of object recognition is achieved through the use of 517 images of 10 objects consisting of pedestrians, cars, bicycles and 7 traffic signs, and of convolutional neural networks models including SSD Inception V2, Faster R-CNN Inception V2, Faster R-CNN Resnet 50 and Faster R-CNN Resnet 101, which are known as the basis of deep learning. The models previously trained on the COCO data set are retrained and evaluated on the new data set with the transfer learning method. The new data set is formed by part of the image from the GRAZ-01 and GRAZ-02 data sets and part of the image from the mobile phone camera.

### 2.3 Smart City Lane Detection for Autonomous Vehicle

Dawam et al developed a computer vision based road surface marking recognition system to serve as an added layer of data source from which AVs will make decisions. We trained our detector using YOLOv3 running in the cloud to detect 25 classes of Road surface markings using over 25,000 images.

### 2.4 Traffic Sign Recognition Based on Improved SSD Model

Huo et al presented an improved SSD model for traffic road sign recognition was proposed. The model uses DenseNet to replace SSD's basic network VGG16 to reduce the amount of network parameters; learn from the feature fusion method of FCN to improve the detection ability of small targets; increase the convolution of holes to expand the perception domain and reduce the loss of small target information; use depth Separable convolution replaces the Maxpooling layer in DenseNet to avoid information loss during feature extraction. Experiments were carried out on the CTSD data set.

### 2.5 Traffic Sign Detection And Recognition Using TensorFlow' s Object Detection API With A New Benchmark Dataset

Kilic et al implemented a TensorFlow's Object Detection API used in Turkey traffic sign detection and recognition of the aid intended traffic signs. A dataset suitable for TensorFlow's Object Detection API has been prepared for traffic signs by using photographs in different traffic and weather conditions. A web-based data tagging application has been developed to prepare this dataset.

### 2.6 On-The-Fly Traffic Sign Image Labeling

Jeya et al provided the driving device with precise information. The Dataset that is being used are a collection of traffic signs. Using the darknet module we set up YOLO(You Only Look Once). Yolo is a predefined CNN which is used to label objects automatically. YOLO works on a dataset called the COCO dataset which consists of data from 80 different classes.

### 2.7 Traffic Sign Recognition With Lightweight Two-Stage Model in Complex Scenes

Wang et al presented a novel and flexible two-stage approach. It combines a lightweight superclass detector with a refinement classifier. The main contributions lie in three aspects: (1) We use locations and sizes of signs as prior knowledge to establish a probability distribution model. It can significantly decrease the search range of signs and improve the processing speed, as well as reducing false detection. (2) We propose a high-performance lightweight superclass detector. We introduce the Inception and Channel Attention, by generating multi-scale receptive fields and adaptively adjusting channel features. It alleviates the large scale variance challenge of objects and the interference of background information. Meanwhile, we present a merging Batch Normalization and multi-scale testing method to further improve detection performance. (3) We propose a refinement classifier based on similarity measure learning for the subclass classification. It increases the precision of discriminating similar subclasses and also improves the extensibility of our approach. Our two-stage approach is simple and effective, whose paradigm is different from others.

### 2.8 YOLOv3 Algorithm with additional convolutional neural network trained for traffic sign recognition

Novak et al presented a software based on YOLO that is extended with a CNN for traffic sign recognition. Since real time detection is required for safe driving, YOLO network used in this paper is pre trained for detection and classification of only five objects which are separated in categories such as cars, trucks, pedestrians, traffic signs, and traffic lights. Detected traffic signs are further passed to CNN which can classify them in one of 75 categories.

### 2.9 Traffic Signs Recognition in a mobile-based application using TensorFlow and Transfer Learning technics

Benhamida et al implemented a powerful tool for traffic signs recognition in a mobile-based application. This tool uses TensorFlow together with transfer learning technic that makes it easier to train our dataset on a pre-trained Model using the convolutional network (ConvNet). The used model is a Single Shot MultiBox Detector (SSD) MobileNet V2 based model which uses one single deep network to train the model on multiple objects per image. This network uses

300x300 annotated input images with multiple objects to provide faster training time and faster detection results compared to other types of neural networks. The annotation is made by providing the coordinates of the rectangle that surrounds the given object together with its label which defines the name of the object. The coordinates are usually given by providing the (x,y) coordinates of the top-left and bottom-right points of the surrounding rectangle. This presents a powerful technic for real-time detection on mobile devices with low computational capabilities.

**2.10 Traffic sign recognition by combining global and local features based on semi-supervised classification**

He et al described a novel semi-supervised learning approach combining global and local features for TSR in an IOT-based transport system. In their approach, histograms of oriented gradient, colour histograms (CH), and edge features (EF) are used to build different feature spaces. Meanwhile, on the unlabelled samples, a fusion feature space is found to alleviate the differences between different feature spaces.

**2.11 Real-Time Traffic Sign Detection and Recognition using CNN**

Santos et al developed on streets and highways have a distinct set of features which may be used to differentiate each one from each other. We propose in this paper a real-time traffic sign detection and recognition algorithm using neural networks. In order to detect traffic sign we used a Faster R-CNN (Region-Based Convolutional Neural Network), and to classify we used a Convolutional Neural Network using two different architectures. Some factors can make it difficult, such as light, occlusion, blurring, and others. This work can be applied in several areas, such as Advanced Driving Assistant System and autonomous cars.

**2.12 Traffic Sign Detection and Recognition Using Novel Center-Point Estimation and Local Features**

Wei et al developed traffic sign detection as a region classification problem and propose a two-stage CNN-based approach to solve it. At the first stage, we design an efficient network which is built with improved fire-modules to generate object proposals quickly. The network up-samples and merges the feature maps of different scales to attain a high-resolution fused feature map which contains semantically strong features of multi-scale objects. Specially, the prediction

is made on the fuse feature map and based on the novel center-point estimation. With the overall designs, our region proposal network can achieve high recall value while using low-resolution images. At the second stage, a separate classification network is proposed. The bottleneck of the classification performance is generally caused by the greatly similar appearances between traffic signs. Therefore, we further explore local regions with critical differences between traffic signs to obtain fine-grained local features which help to improve classification. Finally, we evaluate our method on a challenge benchmark Tsinghua-Tencent 100K which provides many large images with small traffic sign instances.

# CHAPTER-3

## EXISTING SYSTEM

This method tackle the traffic sign detection problem using the state-of-the-art of multi-object detection systems such as Faster Region Convolutional Neural Networks (F-RCNN) and Single Shot MultiBox Detector (SSD) combined with various feature extractors such as MobileNet v1 and Inception v2, and also Tiny-YOLOv2. However, the focus of this paper is going to be F-RCNN Inception v2 and Tiny YOLO v2 as they achieved the best results.

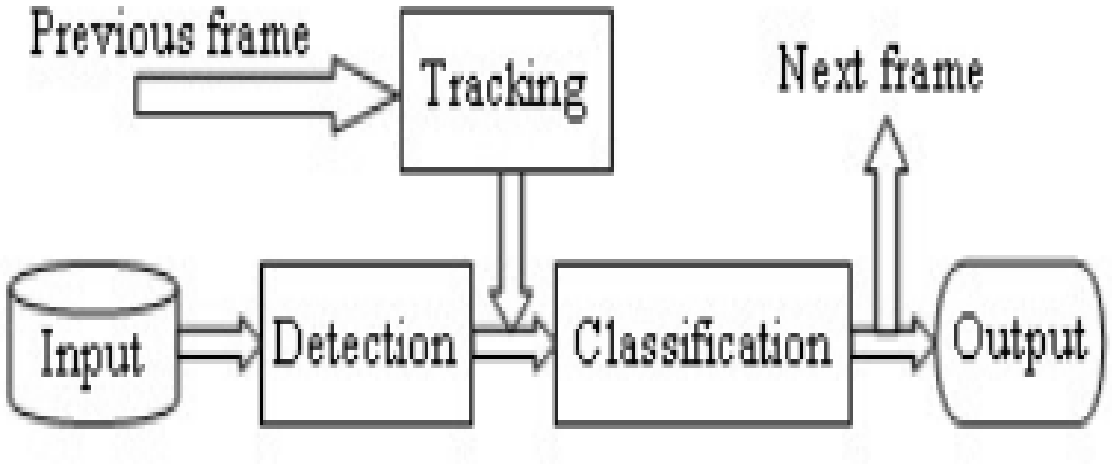



**Fig:3.1 procedure for object detection**

The traffic sign recognition consists of three stages - detection, tracking and recognition as shown in fig. 3.1.

I. Detection The goal of the detection phase is to locate the regions of interest (RoI) in which the object is most likely to be found and indicate the object's presence. During this phase the image is segmented, a potential object is then proposed according to previously provided attributes such as color and shape.

II. Tracking In order to assure the correctness of the proposed region, a tracking phase is needed. Instead of detecting the image using only one frame, the algorithm would track the proposed object for a certain number of frames (usually four). This has proven to increase the accuracy significantly. The most common object tracker is the Kalman Filter.

III. Recognition The recognition phase is the main phase in which the sign is classified to its respective class. Older object recognition techniques may include statistical-based methods, Support Vector Machine, Adaboost and Principal Component Analysis.

**TRAFFIC SIGNS DETECTION AND RECOGNITION**

I. Dataset

Training and testing a Deep Convolutional Neural Network requires a large amount of data as a base. The German Traffic Sign Detection Benchmark (GTSDB) has become the de facto of training Deep CNNs when it comes to traffic sign detection. It includes many types of traffic signs in extreme conditions— weathering, lightening, angles, etc… which help the model train to recognize the signs found in those conditions. The GTSDB contains a total of 900 images (800 for training and 100 for testing). However, this number is clearly not enough for large-scale DCNN models such as F-RCNN Inception.

## II. System Design

### 1. Training phase

First, the training images are loaded in RGB mode, they are then converted to HSV color space. Each image is then passed to the neural network for training. Finally, the network predicts where the traffic sign is (RoI extraction) followed by nonmaximum suppression to choose only the RoIs with the highest confidence, then the model predicts to which class the signs belong. These predictions are then compared to the groundtruth (actual) regions of interest and class labels. The loss function is computed i.e. how far the model was from the correct prediction and back-propagation is then applied to decrease the loss value. This process is repeated for a selected number of epochs, after that the training phase is said to be finished.

### 2. Testing phase

In the testing phase, the images (or video frames) are loaded in RGB mode and then converted to HSV as well, but there is no training, the model just predicts the location and class of the sign.

## III. Network Structure

Various models have been trained and tested, but in this section, the F-RCNN Inception v2 and YOLO v2 models are presented since they produced the best overall results.

Existing speed limit detection and motor speed regulation systems suffer from several drawbacks, which can limit their effectiveness in regulating motor speeds and ensuring road safety. Some of the drawbacks are:

Limited accuracy: Existing speed limit detection systems can suffer from limited accuracy, especially in adverse weather conditions or in low light situations. This can result in incorrect speed limit detection, which can lead to inaccurate motor speed regulation.

High cost: Some existing speed limit detection and motor speed regulation systems can be costly to implement, which can make them inaccessible to low-income communities or regions with limited resources.

Limited flexibility: Some existing systems can be inflexible, meaning that they may not be able to adapt to changes in speed limits or road conditions. This can result in the system being ineffective in certain situations.

Complex installation: Some existing systems can be complex to install and maintain, which can make them difficult to implement in certain regions or for certain types of motors.

Limited features: Some existing systems may only provide speed limit detection and motor speed regulation, without offering additional features such as automatic braking systems or collision detection systems.

Addressing these drawbacks can lead to the development of more effective and accessible speed limit detection and motor speed regulation systems that can help improve road safety and reduce the number of accidents caused by speeding.

# CHAPTER-4

# PROPOSED SYSTEM

The proposed system is a combination of a Python-based speed limit detection system and an Arduino-based motor speed regulation system. The aim of this system is to regulate the speed of a motor based on the speed limit of the area in which the motor is being used. The system consists of two main parts:

## 4.1 Speed Limit Detection System:

The speed limit detection system is implemented using Python and OpenCV. The system uses a camera to capture the video feed of the area in which the motor is being used. The captured video is then processed using the OpenCV library to detect the speed limit signs in the video. The system uses machine learning algorithms to detect and classify the speed limit signs. Once the speed limit is detected, the system sends the speed limit data to the Arduino board for further processing.

## 4.2 Motor Speed Regulation System:

The motor speed regulation system is implemented using an Arduino board and a motor control module. The system receives the speed limit data from the speed limit detection system and uses it to regulate the speed of the motor. The motor control module is connected to the Arduino board, which allows the system to control the speed of the motor. The system uses a Arduino controller to regulate the speed of the motor based on the speed limit. The Arduino controller takes into account the current speed of the motor and the speed limit and adjusts the motor speed accordingly.

The proposed system is designed to be a cost-effective solution for speed limit regulation in areas where speed limits frequently change. The system can be installed on any motor, including cars, trucks, and motorcycles. The system can be customized to work with different types of motors and speed limits. Additionally, the system can be extended to include other features, such as automatic braking systems, collision detection systems, and traffic signal

detection systems.

The proposed system provides a simple and effective solution for speed limit regulation in areas where speed limits frequently change. The system can help reduce the number of accidents caused by speeding and improve road safety. The proposed system is a cost-effective solution that can be easily implemented on different types of motors.

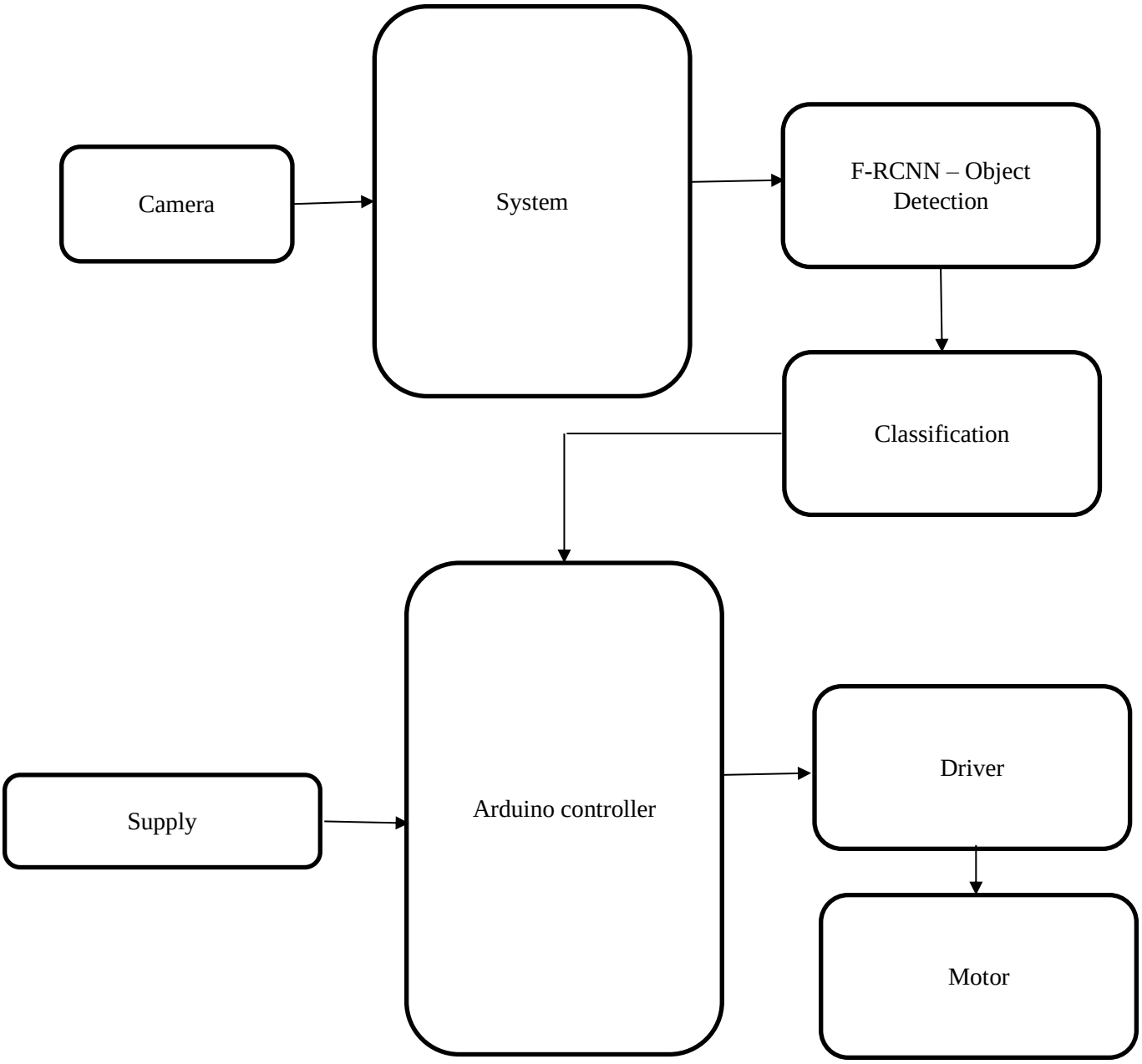



**Fig:4.1 Proposed system**

## 4.3 IMPLEMENTATION :

The working of the proposed system of Python-based speed limit detection and Arduino-based motor speed regulation can be described in the following steps:

**Image capture**: The system uses a camera to capture images of the road ahead.

**Image processing**: The captured images are processed using computer vision techniques to detect and classify speed limit signs. This is done using machine learning algorithms that have been trained on a dataset of speed limit signs.

**Speed limit detection**: Once the speed limit signs are detected and classified, the system determines the speed limit of the area in which the motor is being used.

**Motor speed regulation**: The system uses a PID controller to regulate the speed of the motor based on the detected speed limit. The PID controller calculates the error between the desired speed and the current speed of the motor and adjusts the motor speed accordingly.

**Motor control**: The adjusted motor speed is sent to the Arduino microcontroller, which controls the motor speed.

**Feedback loop**: The system continuously monitors the speed of the motor and adjusts it based on changes in the speed limit. This ensures that the motor speed is always within the speed limit of the area in which it is being used.

The system uses a combination of image processing, machine learning, and PID control techniques to accurately detect speed limit signs and regulate motor speed based on the detected speed limit. The system provides a cost-effective and flexible solution for speed limit regulation that can improve road safety and reduce the number of accidents caused by speeding.

The working of the proposed system of Python-based speed limit detection and Arduino-based motor speed regulation can be implemented using the following Python packages and Arduino pin connections:

Python Packages:

**OpenCV**: for image processing and computer vision techniques

**TensorFlow or Keras**: for machine learning algorithms and model training

**PySerial**: for communication between Python and Arduino

**NumPy**: for numerical calculations and data handling

**Matplotlib**: for data visualization

**Arduino Pin Connections**:

Motor driver module (L298N) connected to the Arduino for motor speed control

Speed sensor (encoder or Hall sensor) connected to the Arduino for speed measurement

Bluetooth module (HC-05 or HC-06) connected to the Arduino for communication with the Python program designed to analyze and classify images. These networks are typically trained using supervised learning techniques, where large datasets of labeled images are used to teach the network to recognize different objects or patterns within images.

The working of a neural network for image detection can be described in the following steps:

Data preparation: A large dataset of labeled images is collected, which includes both positive and negative examples of the objects or patterns to be detected.

Network architecture design: The network architecture is designed based on the requirements of the image detection task. This includes selecting the number and type of layers, the activation functions, and the loss function used during training.

Training: The neural network is trained using the labeled dataset, with the goal of minimizing the difference between the predicted outputs and the true labels. This is achieved through a process called backpropagation, where the error is propagated back through the network to adjust the weights and biases of the neurons.

Testing and validation: Once the network is trained, it is tested on a separate dataset to evaluate its performance. This includes measuring metrics such as accuracy, precision, and recall.

Deployment: The trained neural network is deployed to perform image detection tasks in real-world scenarios. This may involve integrating the network with other systems or hardware, such as cameras or sensors.

# CHAPTER-5

# SOFTWARE AND HARDWARE REQUIREMENTS AND SPECIFICATION

## 5.1 SOFTWARE REQUIRED:

- Python 3.7
- Arduino IDE

## 5.2 HARDWARE REQUIRED:

- System : Windows 11 Home
- Microcontroller : Arduino Nano
- Display : LCD Display
- Motor : DC Motor

## 5.3 SOFTWARE SPECIFICATION:

### 5.3.1 PYTHON 3.7 :

Python is a popular programming language that is widely used in Digital Image Processing (DIP). DIP is an area of study that involves the processing of images using mathematical algorithms and techniques. Python 3.7 is particularly well-suited for DIP because of its rich library of image processing tools and algorithms.One of the most popular libraries for image processing in Python is OpenCV. OpenCV is a computer vision library that provides a wide range of functions for image processing and computer vision applications. With the release of Python 3.7, OpenCV was updated to support the latest version of the language. This means that developers can now take advantage of the latest features and improvements in Python 3.7 when working with OpenCV.

Python 3.7 also includes several built-in modules for image processing, such as the Pillow library. Pillow is a Python Imaging Library that provides a wide range of functions for image processing, including image enhancement, filtering, and transformation. With Pillow, developers

can easily manipulate and transform images using Python 3.7.In conclusion, Python 3.7 is an excellent choice for Digital Image Processing, with its rich library of image processing tools and algorithms. OpenCV and Pillow are just two examples of the many libraries and modules available for image processing in Python 3.7. The continued development and improvement of the language make it an exciting time for DIP developers, as new features and capabilities are added to the language with each new release.

### 5.3.2 KERAS :

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation, allowing researchers and developers to quickly prototype and test new neural network architectures. Keras provides a simple and intuitive interface for building and training neural networks, making it an ideal choice for beginners and experts alike. It offers a wide range of pre-built layers, loss functions, and optimizers, as well as the ability to define custom layers and loss functions. Keras also supports both CPU and GPU acceleration for faster training and inference.

### 5.3.3 TENSORFLOW :

TensorFlow is an open-source machine learning framework developed by Google Brain Team. It is a comprehensive ecosystem of tools, libraries, and community resources that enables developers to build and deploy machine learning models at scale. TensorFlow supports a wide range of machine learning tasks, including classification, regression, and clustering, as well as deep learning tasks such as image recognition and natural language processing. TensorFlow provides a low-level API for building custom models and performing complex operations, as well as a higher-level API for building and training neural networks more easily. It also supports distributed training across multiple GPUs and CPUs, making it a powerful tool for large-scale machine learning projects.

### 5.3.4 OPENCV :

OpenCV (Open Source Computer Vision) is an open-source computer vision and machine learning library. It was originally developed by Intel in 1999 and has since become one of the most popular and widely-used computer vision libraries, with over 47 thousand people in the user community. OpenCV provides a wide range of tools and algorithms for image and video processing, object detection and recognition, feature extraction, and more. It supports a variety of programming languages, including Python, C++, and Java, and can be run on a range of operating systems, including Windows, macOS, and Linux. OpenCV is an essential tool for researchers, developers, and engineers who are working on computer vision applications and want to leverage pre-built algorithms and functions to speed up their development process.

### 5.3.5 PYSERIAL :

PySerial is a Python library that allows you to communicate with serial devices, such as Arduino boards, over a serial port. It provides a simple interface for reading and writing data to and from the serial port, as well as setting various parameters like baud rate and parity. PySerial is cross-platform, meaning it can be used on Windows, macOS, and Linux.

### 5.3.6 NUMPY :

NumPy is a Python library for numerical computing. It provides an efficient implementation of multi-dimensional arrays and various mathematical functions for operating on these arrays. NumPy is widely used for scientific computing and data analysis, as it allows for efficient computation on large datasets. Additionally, it includes functionality for linear algebra, Fourier transforms, and random number generation, among other things. Overall, NumPy is a powerful library that enables fast and flexible numerical operations in Python.

### 5.3.7 MATPLOTLIB :

Matplotlib is a Python library for creating visualizations, including charts, graphs, and plots. It provides a simple and customizable interface for creating high-quality visualizations in Python. With Matplotlib, you can create a wide range of visualizations, including line plots, scatter plots, bar charts, histograms, and more. The library is highly customizable, allowing you to adjust every aspect of your visualization, from the size and position of the plot to the color and style of the lines or markers. Matplotlib is widely used in scientific computing, data analysis, and machine learning, making it an essential tool for any data scientist or analyst working in Python.

### 5.3.8 Arduino IDE :

| Arduino Software IDE | |
|---|---|
|  A screenshot of the Arduino IDE showing the "Blink" program, a simple beginner program | |
| Developer(s) | Arduino Software |
| Stable release | 1.6.7 / 17 December 2015; 49 days ago[17] |
| Written in | Java, C and C++ |
| Operating system | Cross-platform |
| Type | Integrated development environment |
| License | LGPL or GPL license |
| Website | arduino.cc |

**Fig:5.1 A screenshot of the Arduino IDE showing the "Blink" program**

Arduino programs may be written in any programming language with a compiler that produces binary machine code. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in Java. It originated from the IDE for the Processing programming language project and the Wiring project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism for compiling and loading programs to an Arduino board. A program written with the IDE for Arduino is called a "sketch".

The Arduino IDE supports the C and C++ programming languages using special rules of code organization. The Arduino IDE supplies a software library called "Wiring" from the Wiring project, which provides many common input and output procedures. A typical Arduino C/C++ sketch consist of two functions that are compiled and linked with a program stub main () into an executable cyclic executive program:

Setup (): a function that runs once at the start of a program and that can initialize settings.

Loop (): a function called repeatedly until the board powers off.

After compilation and linking with the GNU toolchain, also included with the IDE distribution, the Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal coding that is loaded into the Arduino board by a loader program in the board's firmware.

## 5.4 HARDWARE SPECIFICATION:

### 5.4.1 ARDUINO MICROCONTROLLER :

Arduino is an open-source computer hardware and software company, project and user community that designs and manufactures microcontroller-based kits for building digital devices and interactive objects that can sense and control objects in the physical world.

The project is based on microcontroller board designs, manufactured by several vendors,

using various microcontrollers. These systems provide sets of digital and analog I/O pins that can be interfaced to various expansion boards ("shields") and other circuits. The boards feature serial communications interfaces, including USB on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino project provides an integrated development environment (IDE) based on the Processing project, which includes support for the C and C++ programming languages.

The first Arduino was introduced in 2005, aiming to provide an inexpensive and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

Arduino boards are available commercially in preassembled form, or as do-it-yourself kits. The hardware design specifications are openly available, allowing the Arduino boards to be manufactured by anyone. AdafruitIndustriesestimated in mid-2011 that over 300,000 official Arduinos had been commercially produced, and in 2013 that 700,000 official boards were in users' hands.

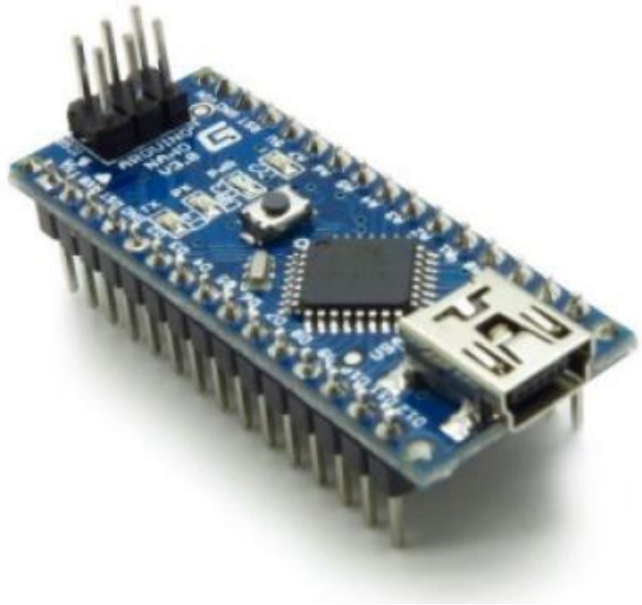
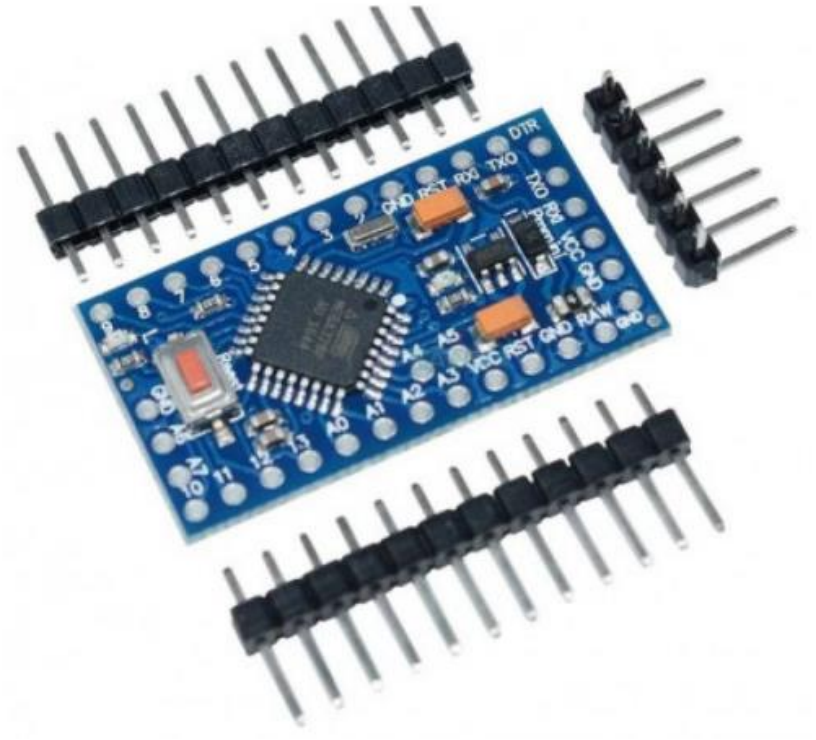


**Fig 5.2: Arduino Nano**




**Arduino Nano**

An early Arduino board with anRS-232 serial interface (upper left) and an Atmel ATmega328 microcontroller chip (black, lower right); the 14 digital I/O pins are located at the top and the six analog input pins at the lower right.

An Arduino board historically consists of an Atmel 8-, 16- or 32-bit AVR microcontroller (although since 2015 other makers' microcontrollers have been used) with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which lets users connect the CPU board to a variety of interchangeable add-on modules known as shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus—so many shields can be stacked and used in parallel. Prior to 2015 Official Arduinos had used the Atmel megaAVR series of chips, specifically the ATmega8,ATmega168, ATmega328, ATmega1280, and ATmega2560 and in 2015 units by other manufacturers were added. A handful of other processors have also been used by Arduino compatibles. Most boards include a 5 V linear regulator and a 16 MHz crystal

oscillator (or ceramic resonator in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer. This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer. Currently, opti boot boot loader is the default boot loader installed on Arduino UNO.

At a conceptual level, when using the Arduino integrated development environment, all boards are programmed over a serial connection. Its implementation varies with the hardware version. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and TTL-level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the FTDI FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods, when used with traditional microcontroller tools instead of the Arduino IDE, standard AVR ISP programming is used.

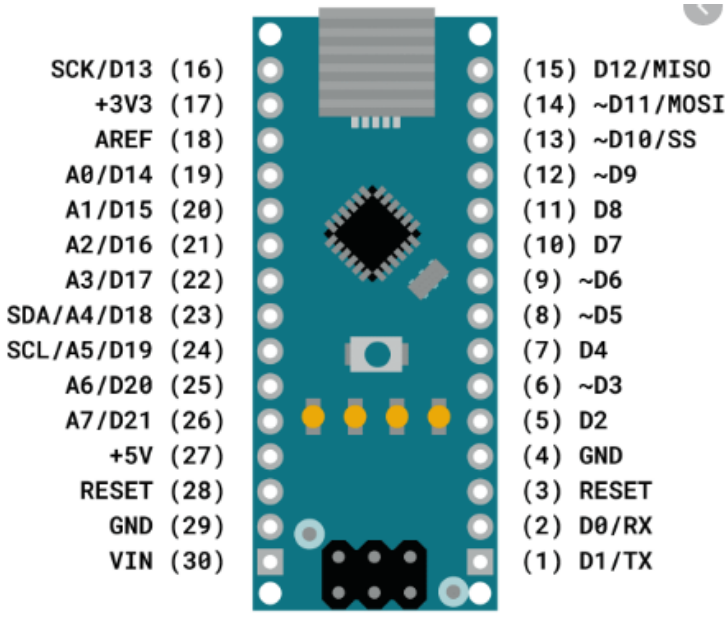



**Fig: 5.3 An official Arduino Uno with descriptions of the I/O locations**

The Arduino board exposes most of the microcontroller's I/O pins for use by other circuits. The Diecimila, Duemilanove, and current Uno provide 14 digital I/O pins, six of which can produce pulse-width modulated signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.10-inch (2.5 mm) headers. Several plug-in application shields are also commercially available. The ArduinoNano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into solderless breadboards.

There are many Arduino-compatible and Arduino-derived boards. Some are functionally equivalent to an Arduino and can be used interchangeably. Many enhance the basic Arduino by adding output drivers, often for use in school-level education to simplify the construction of buggies and small robots. Others are electrically equivalent but change the form factor, sometimes retaining compatibility with shields, sometimes not. Some variants use completely different processors, with varying levels of compatibility.

### 5.4.2 LCD – Liquid Crystal Display

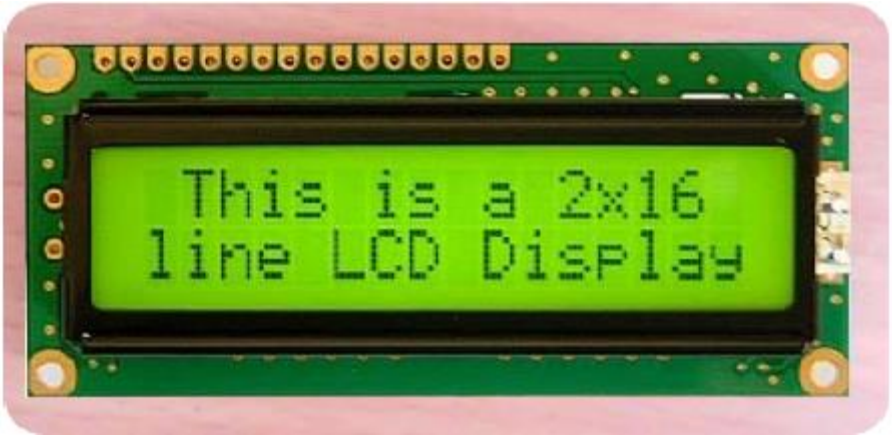



**Fig: 5.4 LCD**

Liquid Crystal Displays (LCDs) have materials, which combine the properties of both liquid and crystals. Rather than having a melting point, they have a temperature range within which the

molecules are almost as mobile as they would be in a liquid, but are grouped together in an ordered form similar to a crystal. An LCD consists of two glass panels, with the liquid crystal material sand witched in between them. The inner surface of the glass plates are coated with transparent electrodes which define the character, symbols or patterns to be displayed polymeric layers are present in between the electrodes and the liquid crystal, which makes the liquid crystal molecules to maintain a defined orientation angle. One each polarizer are pasted outside the two glass panels. This polarizer would rotate the light rays passing through them to a definite angle, in a particular direction. When the LCD is in the off state, light rays are rotated by the two polarizer and the liquid crystal, such that the light rays come out of the LCD without any orientation, and hence the LCD appears transparent. When sufficient voltage is applied to the electrodes, the liquid crystal molecules would be aligned in a specific direction. The light rays passing through the LCD would be rotated by the polarizer, which would result in activating / highlighting the desired characters.

The LCDs are lightweight with only a few millimeters thickness. Since the LCD's consume power, they are compatible with low power electronic circuits, and can be powered for long durations. The LCD does don't generate light and so light is needed to read the display. By using backlighting, reading is possible in the dark. The LCD's have long life and a wide operating temperature range. Changing the display size of the layout size is relatively simple which makes the LCD's more customers friendly. The LCD's used exclusively in watches, calculators and measuring instruments are the simple seven-segment displays, having a limited amount of numeric data. The recent advances in technology have resulted in better legibility, more information displaying capability and a wider temperature range. These have resulted in the LCDs being extensively used in telecommunications and entertainment electronics. The LCDs have even started replacing the cathode ray tubes (CRTs) used for the display of text and graphics, and also in small TV applications.

LCD display use of our project title message and information message. Our project connect to a microcontroller unit data line connected to a 'PORT 2' and control lines connected to a P3.5,P3.6,P3.7.

### 5.4.3 DC MOTOR :

A DC motor is any of a class of rotary electrical motors that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

I. DC motors were the first form of motor widely used, as they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight brushed motor used for portable power tools and appliances. Larger DC motors are currently used in propulsion of electric vehicles, elevator and hoists, and in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications.

II. A coil of wire with a current running through it generates an electromagnetic field aligned with the center of the coil. The direction and magnitude of the magnetic field produced by the coil can be changed with the direction and magnitude of the current flowing through it.

III. A simple DC motor has a stationary set of magnets in the stator and an armature with one or more windings of insulated wire wrapped around a soft iron core that concentrates the magnetic field. The windings usually have multiple turns around the core, and in large

**Fig: 5.5 DC MOTOR**

motors there can be several parallel current paths. The ends of the wire winding are connected to a commutator. The commutator allows each armature coil to be energized in turn and connects the rotating coils with the external power supply through brushes. (Brushless DC motors have electronics that switch the DC current to each coil on and off and have no brushes.)

IV. The total amount of current sent to the coil, the coil's size and what it's wrapped around dictate the strength of the electromagnetic field created.

V. The sequence of turning a particular coil on or off dictates what direction the effective electromagnetic fields are pointed. By turning on and off coils in sequence a rotating magnetic field can be created. These rotating magnetic fields interact with the magnetic fields of the magnets (permanent or electromagnets) in the stationary part of the motor (stator) to create a torque on the armature which causes it to rotate. In some DC motor designs the stator fields use electromagnets to create their magnetic fields which allow greater control over the motor.

VI. At high power levels, DC motors are almost always cooled using forced air.

VII. Different number of stator and armature fields as well as how they are connected provide different inherent speed/torque regulation characteristics.

VIII. The speed of a DC motor can be controlled by changing the voltage applied to the armature. The introduction of variable resistance in the armature circuit or field circuit allowed speed control. Modern DC motors are often controlled by power electronics systems which adjust the voltage by "chopping" the DC current into on and off cycles which have an effective lower voltage.

IX. Since the series-wound DC motor develops its highest torque at low speed, it is often used in traction applications such as electric locomotives, and trams. The DC motor was the mainstay of electric traction drives on both electric and diesel-electric locomotives, street-cars/trams and diesel electric drilling rigs for many years. The introduction of DC motors and an electrical grid system to run machinery starting in the 1870s started a new second Industrial Revolution. DC motors can operate directly from rechargeable batteries,

providing the motive power for the first electric vehicles and today's hybrid cars and electric cars as well as driving a host of cordless tools. Today DC motors are still found in applications as small as toys and disk drives, or in large sizes to operate steel rolling mills and paper machines. Large DC motors with separately excited fields were generally used with winder drives for mine hoists, for high torque as well as smooth speed control using thyristor drives. These are now replaced with large AC motors with variable frequency drives.

x. If external mechanical power is applied to a DC motor it acts as a DC generator, a dynamo. This feature is used to slow down and recharge batteries on hybrid and electric cars or to return electricity back to the electric grid used on a street car or electric powered train line when they slow down. This process is called regenerative braking on hybrid and electric cars. In diesel electric locomotives they also use their DC motors as generators to slow down but dissipate the energy in resistor stacks. Newer designs are adding large battery packs to recapture some of this energy.

## CHAPTER-6

## RESULT AND DISCUSSION

```
#Training model
model = Sequential()    ## creating a blank model
model.add(Conv2D(32,kernel_size=(3,3),activation='relu',input_shape=(224,224
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))     ### reduce the overfitting

model.add(Flatten())     ### input layer
model.add(Dense(256,activation='relu'))     ## hidden layer of ann
model.add(Dropout(0.5))
model.add(Dense(5,activation='softmax'))    ## output layer

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['acc

model.summary()
```

**Fig: 6.1 Figure Layer designing in python**

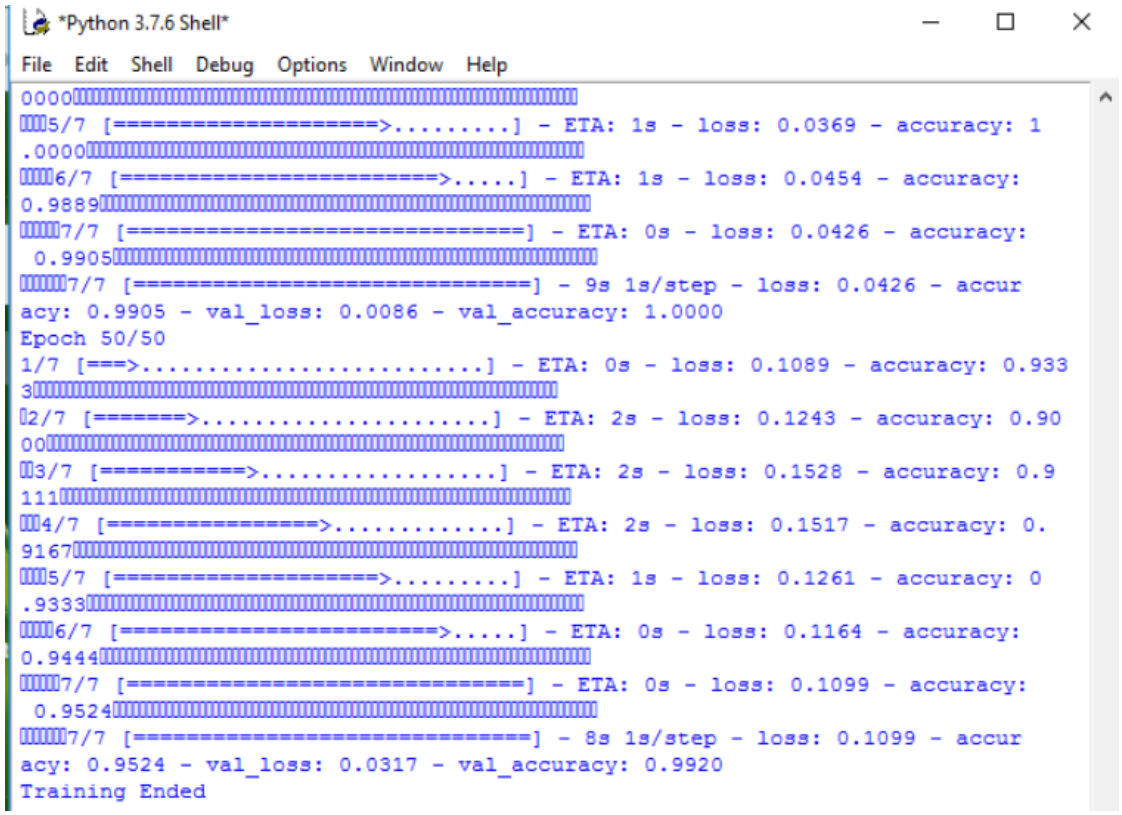



**Fig: 6.2 Figure Training**

```
success
Please give the image name to predict : D:\Project\SLRC (FYproject)\Traffic_sign
 2nd Phase_Arduino\003_0021.png
WARNING:tensorflow:From D:\Project\SLRC (FYproject)\Traffic_sign 2nd Phase_Ardui
no\testing.py:55: Sequential.predict_classes (from tensorflow.python.keras.engin
e.sequential) is deprecated and will be removed after 2021-01-01.
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model doe
s multi-class classification   (e.g. if it uses a `softmax` last-layer activatio
n).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary cla
ssification   (e.g. if it uses a `sigmoid` last-layer activation).
THE Predicted Traffic Sign for the given image is Maintain Speed of 40 km/hr

Please give the image name to predict :
```

**Fig: 6.3 Figure Image with classification**

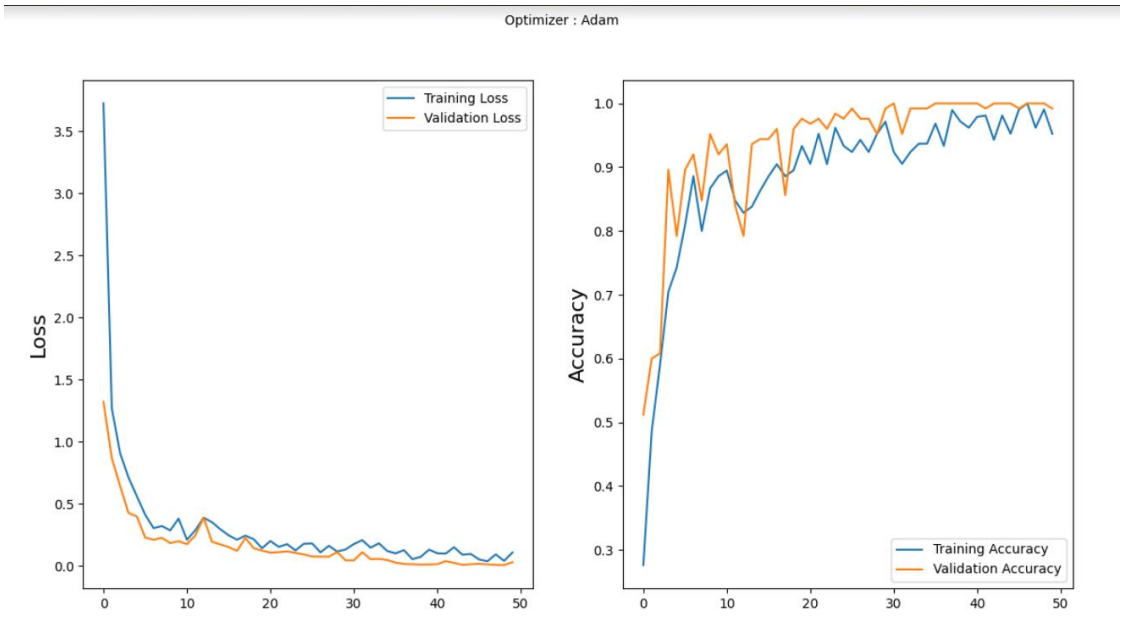



**Fig: 6.4 Figure Performance analysis**

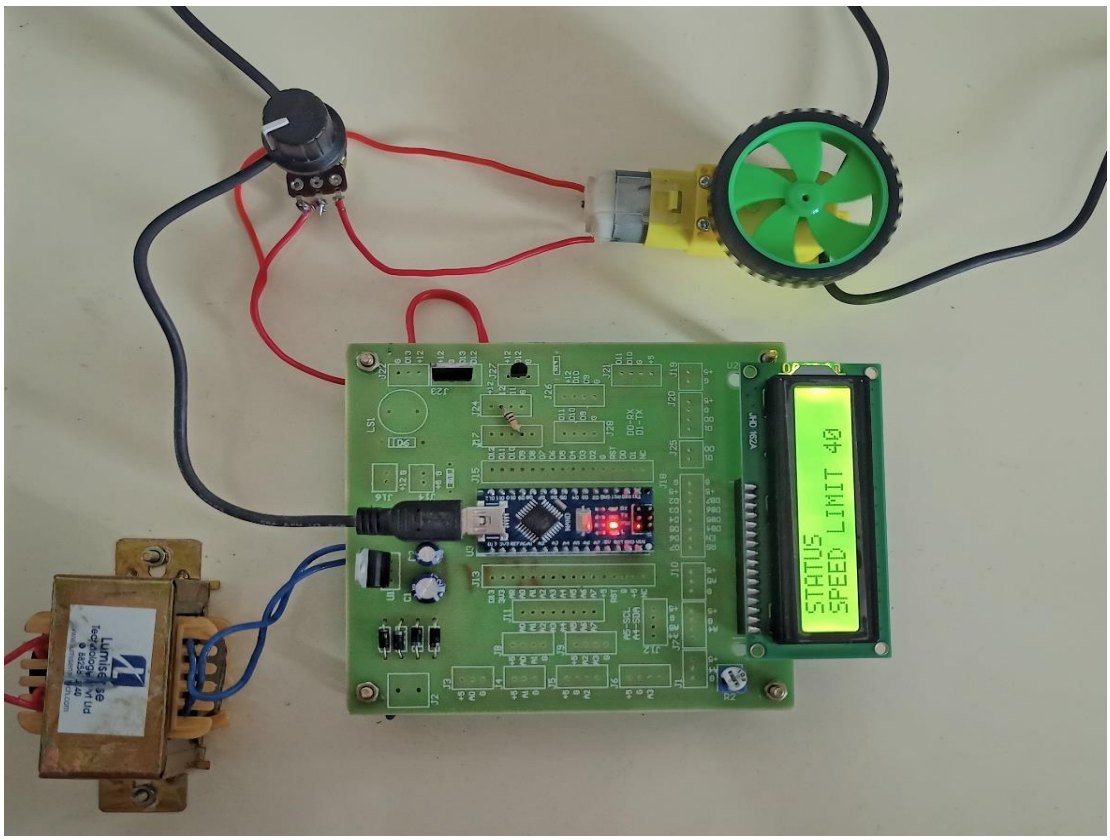



**Fig: 6.5 Figure Hardware Output**

# CHAPTER-7

## CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, the proposed system that combines a Python-based speed limit detection system and an Arduino-based motor speed regulation system has the potential to significantly improve road safety by reducing the number of accidents caused by speeding. By using machine learning algorithms to detect and classify speed limit signs, and an Arduino microcontroller to regulate the speed of the motor based on the speed limit, the system provides a cost-effective and easy-to-implement solution for speed limit regulation in areas where speed limits frequently change. The system can also be extended to include additional features such as automatic braking systems and collision detection systems, further enhancing its potential to improve road safety. Overall, this project demonstrates the power of combining machine learning and microcontroller technologies to address real-world problems and make a positive impact on society.

## REFERENCES

**[1]** L. Wei, C. Xu, S. Li and X. Tu, "Traffic Sign Detection and Recognition Using Novel Center-Point Estimation and Local Features," in IEEE Access, vol. 8, pp. 83611-83621, 2020, doi: 10.1109/ACCESS.2020.2991195.

**[2]** D. C. Santos et al., "Real-Time Traffic Sign Detection and Recognition using CNN," in IEEE Latin America Transactions, vol. 18, no. 03, pp. 522-529, March 2020, doi: 10.1109/TLA.2020.9082723.

**[3]** Z. He, F. Nan, X. Li, S. Lee and Y. Yang, "Traffic sign recognition by combining global and local features based on semi-supervised classification," in IET Intelligent Transport Systems, vol. 14, no. 5, pp. 323-330, 5 2020, doi: 10.1049/iet-its.2019.0409.

**[4]** A. Benhamida, A. R. Várkonyi-Kóczy and M. Kozlovszky, "Traffic Signs Recognition in a mobile-based application using TensorFlow and Transfer Learning technics," 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), Budapest, Hungary, 2020, pp. 000537-000542, doi: 10.1109/SoSE50414.2020.9130519.

**[5]** B. Novak, V. Ilić and B. Pavković, "YOLOv3 Algorithm with additional convolutional neural network trained for traffic sign recognition," 2020 Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, Serbia, 2020, pp. 165-168, doi: 10.1109/ZINC50678.2020.9161446.

**[6]** Z. Wang, J. Wang, Y. Li and S. Wang, "Traffic Sign Recognition With Lightweight Two-Stage Model in Complex Scenes," in IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2020.3020556.

**[7]** J. Jeya Visshwak, P. Saravanakumar and R. I. Minu, "On-The-Fly Traffic Sign Image Labeling," 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2020, pp. 0530-0532, doi: 10.1109/ICCSP48568.2020.9182075.

**[8]** I. Kilic and G. Aydin, "Traffic Sign Detection And Recognition Using TensorFlow' s Object Detection API With A New Benchmark Dataset," 2020 International Conference on Electrical Engineering (ICEE), Istanbul, Turkey, 2020, pp. 1-5, doi: 10.1109/ICEE49691.2020.9249914.

**[9]** A. Huo, W. Zhang and Y. Li, "Traffic Sign Recognition Based on Improved SSD Model," 2020 International Conference on Computer Network, Electronic and Automation (ICCNEA), Xi'an, China, 2020, pp. 54-58, doi: 10.1109/ICCNEA50255.2020.00021.

**[10]** E. S. Dawam and X. Feng, "Smart City Lane Detection for Autonomous Vehicle," 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Calgary, AB, Canada, 2020, pp. 334-338, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00065.

**[11]** G. ÖZTÜRK, R. KÖKER, O. ELDOğAN and D. KARAYEL, "Recognition of Vehicles, Pedestrians and Traffic Signs Using Convolutional Neural Networks," 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Istanbul, Turkey, 2020, pp. 1-8, doi: 10.1109/ISMSIT50672.2020.9255148.

**[12]** L. Du et al., "Improved detection method for traffic signs in real scenes applied in intelligent and connected vehicles," in IET Intelligent Transport Systems, vol. 14, no. 12, pp. 1555-1564, 12 2020, doi: 10.1049/iet-its.2019.0475.

**[13]** X. Yang, Y. Qu, and S. Fang, “Color fused multiple features for traffic sign recognition,” in Proc. ICIMCS, 2012, pp. 84–87.

**[14]** S. Salti, A. Petrelli, F. Tombari, N. Fioraio, and L. D. Stefano, “Traffic sign detection via interest region extraction,” Pattern Recognit., vol. 48, no. 4, pp. 1039–1049, Apr. 2015.

**[15]** J. Greenhalgh and M. Mirmehdi, “Real-time detection and recognition of road traffic signs,” IEEE Trans. Intell. Transp. Syst., vol. 13, no. 4, pp. 1498–1506, Dec. 2012.

**[16]** G. Overett and L. Petersson, “Large scale sign detection using HOG feature variants,” in Proc. IEEE Intell. Vehicles Symp., Jun. 2011, pp. 326–331.

**[17]** M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, “Traffic sign recognition—How far are we from the solution?” in Proc. IJCNN, Aug. 2013, pp. 1–8.

**[18]** F. Zaklouta and B. Stanciulescu, “Real-time traffic sign recognition in three stages,” Robot. Auto. Syst., vol. 62, no. 1, pp. 16–24, 2014.

**[19]** P. Shopa, Mrs. N. Sumitha, Dr. P.S.K Patra. (2014), „Traffic Sign Detection and Recognition Using OpenCV", International Conference on ICICES2014 - S.A.Engineering College, Chennai, Tamil Nadu, India.

**[20]** Prachi Gawande1,„Traffic Sign Detection and Recognition Using Open CV" International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395 -0056, p-ISSN: 2395-0072, 2017.

**[21]** Dr. D. Y. Patil., „A Road Sign Detection and the Recognition for Driver Assistance Systems" International Conference on Energy Systems and Applications (ICESA 2015).

**[22]** Ching-Hao Lai, Chia-Chen Yu, "An Efficient Real-Time Traffic Sign Recognition System for Intelligent Vehicles with Smart Phones", IEEE International Conference on Technologies and Applications of Artificial Intelligence, 2010.

**[23]** Balluchi, A., Benvenuti, L., Di Benedetto, M.D.,Pinello, C. and Sangiovanni-Vincentelli, A.L.,Automotive engine control and hybrid systems:challenges and opportunities. Proceedings of the IEEE.v88 i7. 888-912.

**[24]** Kassem, N. Microsoft Corp., Redmond, WA, USA Kosba, A.E.; Youssef, M.;VRF-Based Vehicle Detection and Speed Estimation vehicular Technology Conference (VTC Spring), IEEE (2012).

**[25]** Puleston, P.F., Spurgeon, S. and Monsees, G.,Automotive engine speed control: a robust nonlinear control framework. IEE Proceedings - Control Theory and Applications. v148 i1. 81-87

**[26]** O. Dabber et al., "An end-to-end system for crowdsourced 3D maps for autonomous vehicles: The mapping component," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., Sep. 2017, pp. 634–641.