

SMART TRAFFIC CONTROL USING COMPUTER VISION AND OCR VEHICLE MONITORING

A PROJECT REPORT

Submitted by

PRASANNA A (822421106047)

PRITHIVIRAJ P (822421106048)

SANTHOSH B (822421106055)

In partial fulfillment for the degree

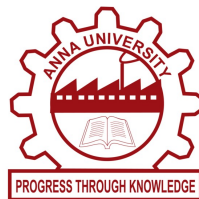
of

BACHELOR OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION ENGINEERING

MRK INSTITUTE OF TECHNOLOGY, KATTUMANNARKOIL



ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“SMART TRAFFIC CONTROL USING COMPUTER VISION AND OCR VEHICLE MONITORING ”** is the bonafide work of **“PRASANNA A (822421106047), PRITHIVIRAJ P(822421106048), SANTHOSH B (822421106055), ”** who carried out the project work under my supervision.

SIGNATURE

Mr.S.SATHYA MOORTHY M.E.,
SMISEE (ECE), MISTE., MIEEE.,
HEAD OF THE DEPARTMENT,
Assistant Professor
Department of ECE,
MRK Institute of Technology,
Kattumannarkoil - 608 301.

SIGNATURE

Mrs.T.KARTHIGA M.E.,
SUPERVISOR,
Assistant Professor,
Department of ECE,
MRK Institute of Technology,
Kattumannarkoil - 608 301.

Submitted for the Project work / Internship (EC3811) and Viva-Voce examination held on _____ at MRK Institute of Technology, Kattumannarkoil, Cuddalore District.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

It is an indeed a great pleasure and proud privilege to acknowledge the help and support we received from the positive minds around us in making the endeavor successful one. First and foremost we express my sincere gratitude to the almighty.

The infrastructural support with all kinds of lab facilities have been a motivating factor in the completion of work, all because of our beloved **Chairman Mr.M.R.K.P.Kathiravan B.E.**, with great pleasure we take the opportunity to thank him.

From the academic side the constant support from our respected **Principal Dr.K.Anandavelu B.E., M.E., Ph.d.**, who has encouraged us to work hard to complete the project.

Our sincere thanks to our **Head of the Department and Project Guide Mrs.T.KARTHIGA M.E., and Project coordinator Mr.S.SathyaMoorthy M.E., SMISEE (ECE)., MISTE., MIEEE., Assistant Professor** who have given us both moral and technical support which added the experience to the job we have undertaken.

We also thank our **Administrative Officer, Manager, Staff Members, Librarian, Non Teaching Staff Members and Our Friends** of our college, who supported and motivated in all our endeavors to complete the project.

PRASANNA A	(822421106047)
PRITHIVIRAJ P	(822421106048)
SANTHOSH B	(822421106055)

ABSTRACT

Traffic congestion, rule violations, and lack of emergency prioritization are significant problems in modern urban environments. Most existing traffic control systems rely on fixed-timer signal operations and manual monitoring, which are outdated and inefficient. They do not offer dynamic control based on real-time conditions and cannot detect violations such as riding without helmets or unauthorized vehicle movement. Our proposed system aims to overcome these drawbacks by integrating Artificial Intelligence (AI), Computer Vision, and Optical Character Recognition (OCR) into a smart traffic management solution. It uses an ESP32-CAM module to capture live video footage at intersections and critical road segments. The system performs real-time helmet detection, vehicle number plate recognition, emergency vehicle detection, and traffic density estimation using machine learning algorithms. OCR technology enables automatic identification of number plates for violation tracking and stolen vehicle detection. The system updates data to a centralized web server, which can be monitored by traffic authorities to take immediate action. Traffic signal timings are adjusted dynamically based on real-time traffic and emergency status to minimize congestion and enhance flow. When an emergency vehicle is detected, the system gives it priority by automatically controlling the traffic lights and clearing the path. This smart, low-cost, and scalable solution significantly improves road safety, law enforcement, and emergency responsiveness. It is ideal for deployment in smart city infrastructures, subways, and high-traffic intersections to build safer and more efficient transportation systems.

Keywords: Smart Traffic, ESP32-CAM, Artificial Intelligence, OCR, Computer Vision, Emergency Detection, Helmet Violation, Automation, Real-Time Monitoring.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF CONTENT	ii
	LIST OF TABLES	v
	LIST OF FIGURES	vi
	ABBREVIATIONS	vii
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
3	SYSTEM ANALYSIS	6
	3.1 EXISTING SYSTEM	6
	3.1.1 LIMITATIONS OF EXISTING SYSTEM	7
	3.2 PROPOSED SYSTEM	7
	3.2.1 MERITS OF PROPOSED SYSTEM	8
4	SYSTEM SPECIFICATION	9
	4.1 SOFTWARE REQUIREMENTS	9
	4.2 HARDWARE REQUIREMENTS	9
	4.3 BLOCK DIAGRAM	10
5	HARDWARE DESCRIPTION	11
	5.1 POWER SUPPLY	11
	5.2 LINEAR POWER SUPPLY	11
	5.3 TRANSFORMER	11
	5.4 RECTIFIER	12
	5.5 REGULATOR	12
	5.8 ESP32 WIFI CAMERA	13
	5.8.1 FEATURES	13
	5.9 MICRO CONTROLLER	14

	5.9.3 PIC16F877A	15
	5.9.5 PERIPHERAL FEATURES	19
	5.10 REGISTER FILE	20
	5.11 MEMORY ORGANIZATION	20
	5.11.1 PROGRAM MEMORY	20
	ORGANIZATION	
	5.11.1 DATA MEMORY ORGANIZATION	21
	5.12 ADDRESSING MODES	22
	5.12.1 DIRECT ADDRESSING	22
	5.12.2 INDIRECT ADDRESSING	22
	5.13 PIC16F87XA DATA EEPROM	23
	5.14 PIC TIMER	24
	5.15 INPUT/OUTPUT PROGRAMMING IN PIC16F877A	25
	5.16 LED PANEL	27
6	SOFTWARE DESCRIPTION	29
	6.1 MPLAB IDE SOFTWARE	29
	6.1.1 MPLAB IDE	31
	6.1.2 Project Creation	31
	6.1.3 Update Source Code	32
	6.1.4 Building Project	32
	6.1.5 Programming The PIC	32
	6.2 PYTHON	32
	6.3 PYTHON NUMPY	33
	6.4 OPEN CV	34
	6.4.1 What Is Open Cv	35
	6.4.3 Why Open Cv Is Used For Computer Vision	37

	6.4.4 Installation Of The Open Cv	38
	6.5 SCIKIT LEARN	39
	6.6 PANDAS	39
	6.6.1 Library Features	40
	6.7 WEB CAMERA	40
	6.8 TENSOR FLOW - INTRODUCTION	41
	6.9 CONVOLUTIONAL NEURAL NETWORK	43
	6.9.1 Layers in CNN	44
7	METHODOLOGY AND OVERVIEW	47
	7.1 METHODOLOGY	47
	7.2 PYTHON PILLOW -OVERVIEW	48
	7.3 PYTHON PILLOW - ENVIRONMENT SETUP	49
	7.4 PYTHON PILLOW USING IMAGE MODULE	50
	7.5 KERAS INTRODUCTION	50
	7.5.1 FEATURES OF KERAS	50
	7.5.2 BENEFITS OF KERAS	51
	7.6 ARTIFICIAL NEURAL NETWORK	51
	7.6.1 SINGLE LAYER PERCEPTRON	52
	7.6.2 MULTI LAYER PERCEPTRON	52
	7.7 OPTICAL CHARACTER RECOGNITION	52
	7.7.1 Implementation of OCR	53
	7.8 PYTESSERACT	53
	7.9 TRAIN THE MODEL	54
	7.10 APPLICATION	55
	7.11 ADVANTAGES	55
9	PROTOTYPE	56
10	CONCLUSION	57
	REFERENCES & APPENDIX	58

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
5.1	Register File Maps Used in PIC16F877A	18

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
4.1	Block Diagram	10
5.1	ESP32 Wi-Fi Camera	13
5.2	Block Diagram of PIC16F877A	16
5.3	Pin Diagram of PIC16F877A	17
5.4	Memory Organization of PIC16F877A	21
5.5	Addressing Mode Diagram of PIC16F877A	23
5.6	LED	27
6.1	Pixel Recognition	36
6.2	RGB	37
6.3	Pip Installation	38
6.4	Command Prompt	38
6.5	TensorFlow Installation	41
6.6	Pooling Layer	45
6.7	Output Layer	46
7.1	Methodology	47
7.2	Installing Pillow Using Pip	49
7.3	Training the Model	54
9.1	Prototype	56

LIST OF ABBREVIATIONS

Abbreviation	Expansion
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CV	Computer Vision
IC	Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things
LED	Light Emitting Diode
ML	Machine Learning
OCR	Optical Character Recognition
OpenCV	Open Source Computer Vision
PC	Personal Computer
PIC	Peripheral Interface Controller
RAM	Random Access Memory
RGB	Red Green Blue (color model)
ROM	Read-Only Memory
SFR	Special Function Register
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
WiFi	Wireless Fidelity

CHAPTER 1

INTRODUCTION

Modern urban environments are increasingly burdened by rising traffic congestion, safety concerns, and inefficient traffic signal operations. Traditional traffic management systems often operate on static signal timings and rely heavily on manual intervention, which can lead to delays, accidents, and ineffective handling of emergency situations. As cities continue to grow, there is a pressing need for intelligent traffic control systems that can adapt in real-time and enhance road safety and traffic flow efficiency.

With the advent of computer vision, machine learning, and Optical Character Recognition (OCR), it is now possible to design advanced traffic monitoring systems capable of detecting violations, analyzing traffic density, and recognizing vehicle number plates with high accuracy. This project leverages these technologies to develop an automated, responsive, and intelligent traffic management system.

The proposed system focuses on three critical areas: helmet detection for rider safety enforcement, number plate recognition to identify stolen or unauthorized vehicles, and dynamic traffic light control based on congestion levels and emergency vehicle detection.

An ESP32-CAM module is integrated into the system to monitor live traffic footage. The camera captures video streams at intersections and road segments, enabling the detection of emergency vehicles, traffic density estimation, and rule violations such as riders without helmets. The data collected is processed using machine learning algorithms to make real-time decisions and adjust traffic light signals dynamically.

By integrating these features, the system aims to support traffic authorities, reduce congestion, enforce safety regulations, and provide faster clearance for emergency services. This contributes to the development of smart, safe, and efficient urban environments.

CHAPTER 2

LITERATUTRE SURVEY

2.1 INTERNET OF THINGS STRATEGIC RESEARCH ROADMAP

Author:O. Vermesan and M. Harrison

Publish:2009

Abstract: Facing the high pollution environment caused by emergencies, it is necessary to collect the flow data of personnel, vehicles and materials accurately in real time. However, the traditional method obviously cannot meet the requirements and needs shall be realized with the help of new technologies. The maturity of Internet of Things technology provides support to solve this key problem . For example, the radio frequency identification technology (RFID) in the Internet of Things technology has the characteristics of long identification distance, strong anti-interference ability, high accuracy and good data security , which can meet the requirements of data collection in emergencies

2.2 GPS AND MAP MATCHING BASED VEHICLE ACCIDENT DETECTION SYSTEM

Author: M. S. Amin, M. A. S. Bhuiyan, M. B. I. Reaz, and S. S. Nasir

Publish:2013

Abstract:First of all, the evacuation of trapped personnel and vehicles in emergencies is very important. On the one hand, it is to better rescue the injured, at the same time, it can also prevent more people from being injured, minimize the adverse impact of the incident, and create good external conditions for the successful disposal of the incident . By distributing radio frequency identification bracelet to evacuees and configuring radio frequency identification tags on vehicles, the flow information of personnel and vehicles can be accurately and quickly collected. Secondly, besides collecting the information of the

evacuated vehicles, it is also necessary to collect the information of the vehicles participating in the rescue. This part of information is sent to the command center of the rescue system after big data processing to provide help for the rescue organizers to allocate vehicles.

2.3 MAX-FLOW RATE PRIORITY ALGORITHM FOR EVACUATION ROUTE PLANNING

Author: D.Guo, C. Gao, W. Ni, and X. Hu

Publish:2016

Abstract:The key component of the emergency traffic response plan is the selection of flow paths for personnel, vehicles and materials. On the premise of safety and short time, transferring as many people and vehicles in the affected site as possible is one of the intuitive standards to measure the feasibility of the plan . With the aid of emergency evacuation paths, those who are in dangerous or affected sites can be evacuated to safe sites or resettlement places as soon as possible, thus minimizing the losses caused by emergencies. At the same time, rescue staff need to rush to the accident site for rescue, and rescue materials need to be transported to the designated place in time.

2.4 DISTRIBUTED DEVICE CACHING FOR INFORMATION UPDATING IN EMERGENCY COMMUNICATIONS

Author:J. Zhou, C. Beard, and Y. Qian

Publish:2018

Abstract:The underdevelopment of information collection technology and the inconvenience of information transmission and distribution will all lead that organizers of emergency response are unable to take correct countermeasures; travelers are unable to obtain timely, correct and useful information to select the optimal path for evacuation; rescue forces are unable to deliver relief supplies to the scene in time. The increasingly severe situation of

urban emergencies puts forward higher requirements for modern urban traffic emergency response system. It is of great significance to study the introduction of Internet of Things technology into the design of traffic emergency response system under urban emergencies to improve the level of emergency response.

2.5 DEEP REINFORCEMENT LEARNING IN TRAFFIC SIGNAL OPTIMIZATION

Author:Guo et al

Publish:2022

Abstract: A multi-agent DRL system that cooperatively controls both traffic light signals and connected autonomous vehicles (CAVs). CoTV aims to reduce travel time, fuel consumption, and emissions, aligning with sustainable development goals. The system's design facilitates efficient coordination between traffic light controllers and CAVs, leading to improved traffic flow and safety.

2.6 AI-BASED ADAPTIVE TRAFFIC SIGNAL CONTROL

Author:Agrahari et al

Publish: 2024

Abstract: The study emphasizes the transition from fixed-time traffic signals to intelligent systems that adapt based on real-time traffic data. The integration of AI techniques, such as machine learning and deep learning, enables these systems to optimize traffic flow, reduce congestion, and enhance road safety. The review highlights various AI models and their effectiveness in different traffic scenarios, underscoring the potential of AI in revolutionizing traffic management.

2.7 MOVEMENT-CENTRIC DEEP REINFORCEMENT LEARNING

Author: shao et al

Publish:2024

Abstract: A novel traffic signal control system that employs movement-centric deep reinforcement learning. By focusing on lane-level control and utilizing the FRAP algorithm, Move Light dynamically adjusts traffic signals based on real-time data. Experimental results from real-world datasets in Cologne and Hangzhou demonstrated significant improvements in queue length, delay, and throughput compared to existing methods.

2.8 ECONOMIC EVALUATION OF SMART TRAFFIC MANAGEMENT SYSTEMS

Author:Yusuf

Publish:2024

Abstract: An Economic evaluation of smart traffic management systems, focusing on their role in reducing carbon emissions. The study combined economic modeling with case studies to assess the cost-benefit ratio of implementing such systems. Findings indicated significant reductions in fuel consumption, travel times, and greenhouse gas emissions, highlighting the financial viability and environmental benefits of smart traffic solutions.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Conventional traffic control systems typically rely on fixed signal timings and limited surveillance capabilities, resulting in inefficiencies and slow response to dynamic road conditions. In most urban areas, traffic signals operate based on pre-programmed schedules that do not account for real-time traffic density or emergencies.

This often leads to unnecessary delays during off-peak hours and prolonged congestion during peak periods. Furthermore, the identification and penalization of traffic rule violators—such as riders without helmets—depend on manual monitoring, which is prone to human error and limited coverage.

Current vehicle identification methods primarily use manual checks or RFID-based systems, which are ineffective for stolen vehicle tracking on a larger scale.

Emergency vehicle prioritization at traffic signals is either absent or limited to systems requiring manual override or radio-frequency-based systems, which are not always reliable or scalable. Overall, these systems lack automation, adaptability, and integration with real-time data, making them unsuitable for the evolving demands of modern traffic environments.

Without intelligent decision-making, these outdated approaches result in increased fuel consumption, longer commute times, and reduced public safety. Hence, there is a critical need to transition from traditional systems to more responsive, intelligent, and automated solutions. Without intelligent decision-making, these outdated approaches result in increased fuel consumption, longer commute times, and reduced public safety. Hence, there is a critical need to transition from traditional systems to more responsive, intelligent, and automated solutions.

3.1.1 Limitations of Existing System

- Dependence on Manual Monitoring
- No Automated Helmet Detection
- Inefficient Emergency Vehicle Handling
- Absence of Automatic Number Plate Recognition (ANPR)
- Limited Scalability and Coverage
- Error-Prone and Time-Consuming Enforcement Methods
- Inability to Analyze Traffic Density Dynamically

3.2 PROPOSED SYSTEM

The proposed system is an intelligent traffic management solution that integrates computer vision, machine learning, OCR, and the ESP32-CAM module to monitor and control urban traffic in real time. The system primarily addresses three key areas: helmet detection, number plate recognition, and dynamic traffic signal control.

Using the ESP32-CAM, live video feeds are captured at intersections to detect motorbike riders and verify helmet usage. In cases of violation, the system uses machine learning algorithms to identify the rider and automatically triggers OCR to extract the vehicle's number plate for enforcement action. Additionally, the same video stream is analyzed to recognize license plates of all passing vehicles, which are then cross-checked with a database to identify stolen or unauthorized vehicles, enabling real-time alerts. For traffic signal control, the ESP32-CAM monitors vehicle density and detects the presence of emergency vehicles such as ambulances or fire trucks. the system uses machine learning algorithms to identify the rider and automatically triggers OCR to extract the vehicle's number plate for enforcement action. Additionally, the same video stream is analyzed to recognize license plates of all passing vehicles, which are then cross-checked with a database to identify stolen or unauthorized vehicles, enabling real-time alerts. For traffic signal control, the

ESP32-CAM monitors vehicle density and detects the presence of emergency vehicles such as ambulances or fire trucks.

Based on this input, the system dynamically adjusts traffic light timings to reduce congestion and ensure immediate clearance for emergency services.

This integrated approach enhances road safety, ensures rule enforcement, and improves the overall efficiency of traffic flow in smart cities.

3.2.1 Merits of Proposed System

- Real-Time Traffic Monitoring and Control
- Automated Helmet Detection for Rider Safety
- Accurate Number Plate Recognition Using OCR
- Dynamic Signal Timing Based on Traffic Density
- Priority Clearance for Emergency Vehicles
- Integration of AI, ML, and Computer Vision
- Use of ESP32-CAM for Live Surveillance
- Enhanced Enforcement Through Automated Alerts
- Reduced Fuel Consumption and Travel Time
- Scalable and Suitable for Smart City Implementation

CHAPTER 4

SYSTEM SPECIFICATION

4.1 SOFTWARE REQUIREMENT

- PYTHON
- TENSORFLOW
- KERAS
- NUMPY
- PILLOW
- SCIPY
- OPENCV
- CONVOLUTIONAL NEURAL NETWORK (CNN)

4.2 HARDWARE REQUIREMENT

- **ESP32-CAM Module** – Compact, Wi-Fi-enabled camera module for live traffic video capture and monitoring.
- **Microcontroller (PIC16F877A)** – Controls system functions, processes input data, and manages output signals.
- **Power Supply Unit** – Provides stable voltage to all components, includes transformer, rectifier, and voltage regulator.
- **Camera/Webcam** – Captures traffic footage for analysis and monitoring.
- **LED Traffic Indication Panels** – Display real-time traffic signal changes (Red, Yellow, Green).
- **Laptop/PC** – Used for programming, processing, and interfacing with the hardware.

- **USB to UART Converter** – Enables communication between the ESP32-CAM and PC.

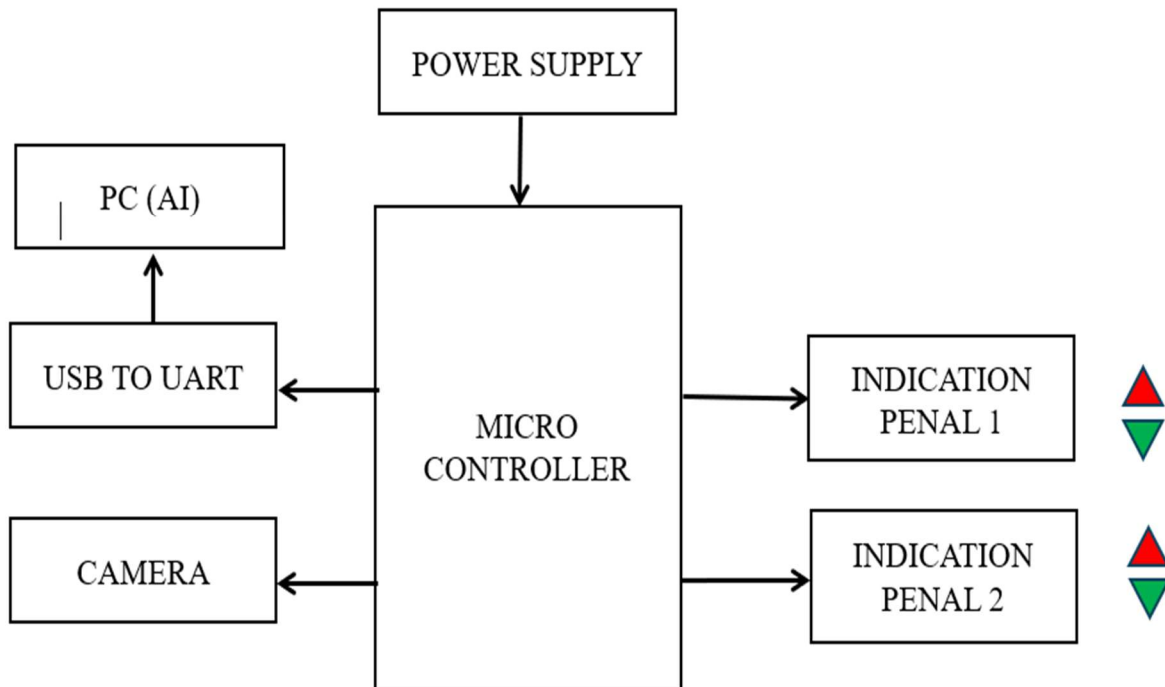


Fig: 4.1 Block Diagram

4.3 Functional Requirements

- Real-time helmet detection using image processing
- Number plate recognition using OCR
- Emergency vehicle detection
- Dynamic traffic signal control
- Automatic data logging and violation alert

CHAPTER 5

HARDWARE DISCRPTION

HARDWARE SPECIFICATION

- System : PC OR LAPTOP
- Processor : INTEL I5
- RAM : 4 GB Recommended
- ROM : 2 GB

5.1 Power Supply

Power supply is a reference to a source of electrical power. A device or system that supplies electrical or other types of energy to an output load or group of loads is called a power supply unit or PSU. The term is most commonly applied to electrical energy supplies, less often to mechanical ones, and rarely to others.

5.2 Linear Power supply:

An AC powered linear power supply usually uses a transformer to convert the voltage from the wall outlet (mains) to a different, usually a lower voltage. If it is used to produce DC, a rectifier is used. A capacitor is used to smooth the pulsating current from the rectifier. Some small periodic deviations from smooth direct current will remain, which is known as ripple. These pulsations occur at a frequency related to the AC power frequency (for example, a multiple of 50 or 60 Hz).

5.3 Transformer:

Transformers convert AC electricity from one voltage to another with little loss of power. Transformers work only with AC and this is one of the reasons why mains electricity is AC.

Step-up transformers increase voltage, step-down transformers reduce voltage. Most power supplies use a step-down transformer to reduce the dangerously high mains voltage (230V in UK) to a safer low voltage.

The input coil is called the primary and the output coil is called the secondary. There is no electrical connection between the two coils; instead they are linked by an alternating magnetic field created in the soft-iron core of the transformer. The two lines in the middle of the circuit symbol represent the core.

5.4 Rectifier:

There are several ways of connecting diodes to make a rectifier to convert AC to DC. The bridge rectifier is the most important and it produces full-wave varying DC. A full-wave rectifier can also be made from just two diodes if a centre-tap transformer is used, but this method is rarely used now that diodes are cheaper. A single diode can be used as a rectifier but it only uses the positive (+) parts of the AC wave to produce half-wave varying DC.

The varying DC output is suitable for lamps, heaters and standard motors. It is not suitable for electronic circuits unless they include a smoothing capacitor.

5.7 Regulator:

Voltage regulator ICs are available with fixed (typically 5, 12 and 15V) or variable output voltages. They are also rated by the maximum current they can pass. Negative voltage regulators are available, mainly for use in dual supplies. Most regulators include some automatic protection from excessive current ('overload protection') and overheating ('thermal protection').

Many of the fixed voltage regulator ICs has 3 leads and look like power transistors, such as the 7805 +5V 1A regulator shown on the right. They include a hole for attaching a heat sink if necessary.

5.8 ESP32 WIFI CAMERA

The ESP32 CAM WiFi Module Bluetooth with OV2640 Camera Module 2MP For Face Recognition has a very competitive small-size camera module that can operate independently as a minimum system with a footprint of only 40 x 27 mm; a deep sleep current of up to 6mA and is widely used in various IoT applications. It is suitable for home smart devices, industrial wireless control, wireless monitoring, and other IoT applications.

5.8.1 Features:

- Flash LED off: 180mA @ 5V.
- Flash LED on to maximum brightness: 310mA @ 5V.
- Deep-sleep: 6mA @ 5V min.
- Modem-sleep: 20mA @ 5V min.
- Light-sleep: 6.7mA @ 5V min.



Fig: 5.1 ESP32 WIFI CAMERA

5.9 MICROCONTROLLER

All the functions required on a single chip. A microcontroller differs from a microprocessor, which is a general-purpose chip that is used to create a multi-function computer or device and requires multiple chips to handle various tasks.

A microcontroller is meant to be more self-contained and independent, and functions as a tiny, dedicated computer.

Early controllers were typically built from logic components and were usually quite large. Later, microprocessors were used, and controllers were able to fit onto a circuit board. Microcontrollers now place all of the needed components onto a single chip. Because they control a single function, some complex devices contain multiple microprocessors. Assembly language consists of short mnemonic descriptions of the instruction sets.

These mnemonics are difficult to remember and the programs developed for one microcontroller cannot be used for other types of microcontrollers.

The most common complaint about microcontroller programming is that the assembly language is somewhat difficult to work with, especially during the development of complex projects.

5.9.1 Evolution of Microcontroller

First, microcontrollers were developed in the mid-1970s. These were basically calculator-based processors with small ROM program memories, very limited RAM data memories, and a handful of input/output ports. As silicon technology developed, more powerful, 8-bit microcontrollers were produced.

In addition to their improved instruction sets, these microcontrollers included on-chip counter/timers, interrupt facilities, and improved I/O handling. On-chip memory capacity was still small and was not adequate for many applications.

One of the most significant developments at this time was the availability of on-chip ultraviolet erasable EPROM memory. This simplified the product development time considerably and, for the first time, also allowed the use of microcontrollers in low-volume applications.

5.9.3 PIC16F877A

The Term Pic, Or Peripheral Interface Controller, Is The Name Given By Microchip Technologies To Its Single – Chip Microcontrollers.

Pic Micros Have Grown To Become The Most Widely Used Microcontrollers In The 8-Bit Microcontroller Segment.

The Pic16F877A Cmos Flash-Based 8-Bit Microcontroller Is Upward Compatible With The Pic16C5x, Pic12Cxxx And Pic16C7x Devices.

It Features 200 Ns Instruction Execution, 256 Bytes Of Eeprom Data Memory, Self Programming, An Icd, 2 Comparators, 8 Channels Of 10-Bit Analog-To-Digital (A/D) Converter, 2 Capture/Compare/Pwm Functions, a Synchronous Serial Port That Can Be Configured As Either 3-Wire Spi Or 2-Wire I2C Bus, a Usart, And a Parallel. The internal block diagram of PIC16F877 is shown in the figure.

It contains 4-banks of register files such as Bank 0, Bank 1, Bank 2 and Bank 3 from 00h-07h, 80h-FFh, 100h-17Fh and 180h-1FFh respectively.

And it is also having program FLASH memory, Data memory and Data EEPROM of 8K, 368 and 256 Bytes respectively.

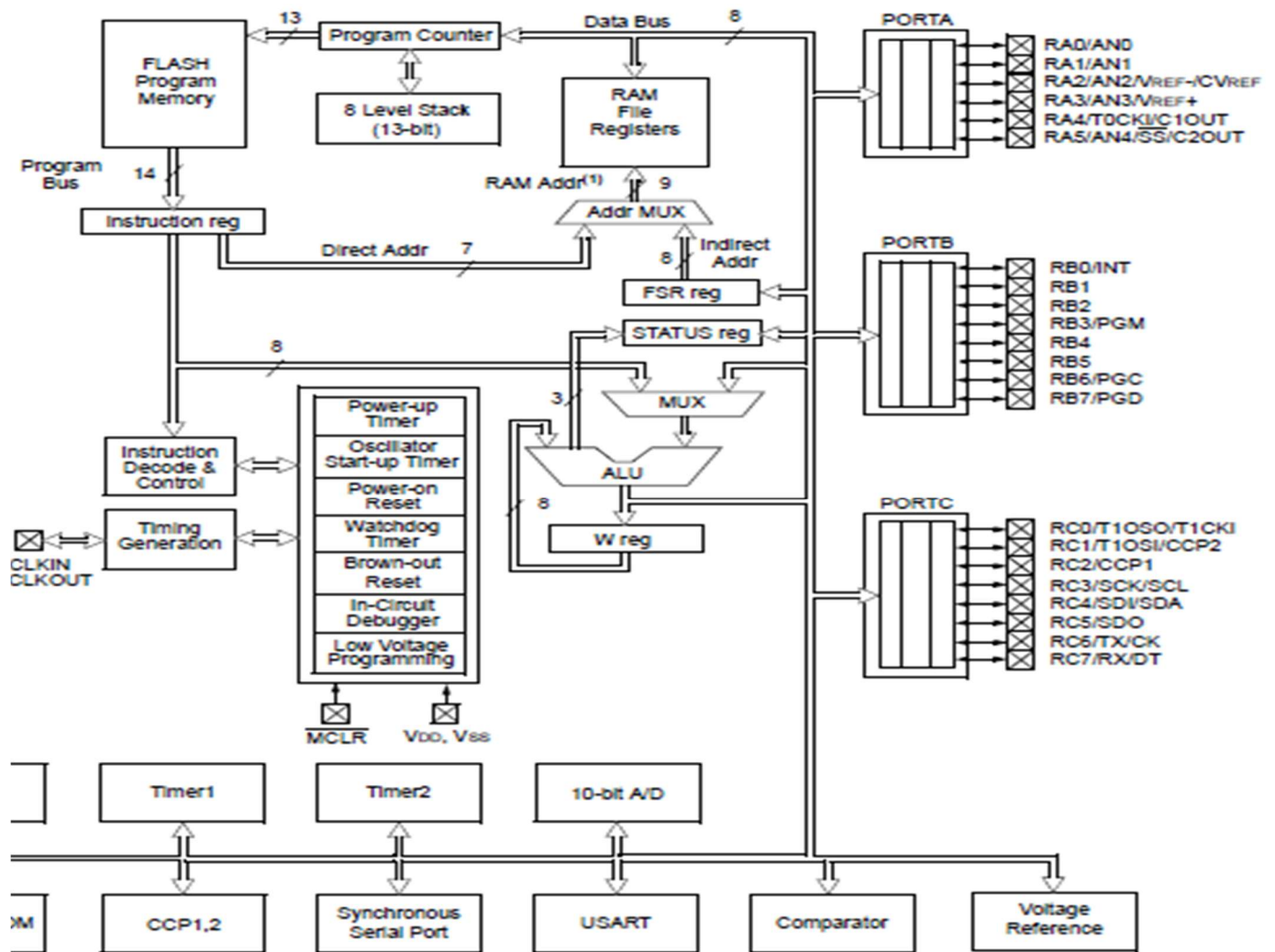


Fig: 5.2 Block Diagram of PIC16F877A

5.9.3.1 Pin Configuration and Description of PIC16F877A

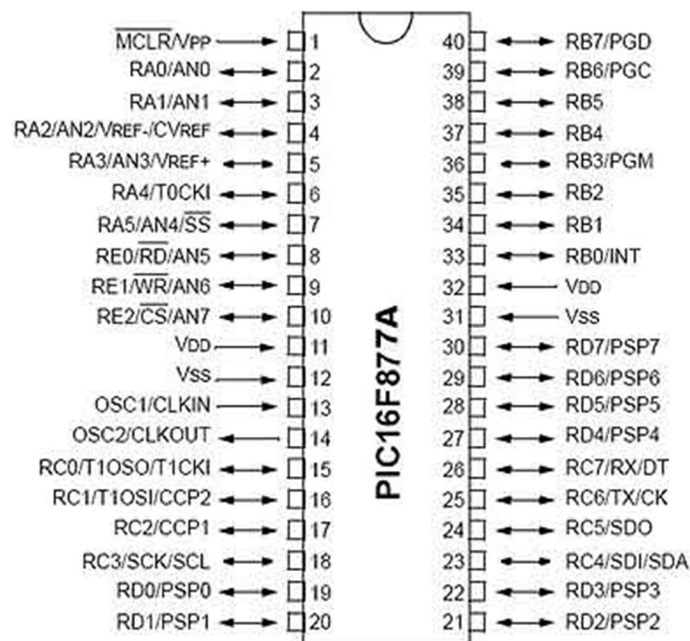


Figure: 5.3 Pin Diagram of PIC16F877A

The pins RB0-RB7, RC0-RC7, and RD0-RD7 are digital I/O pins. The pins CCP1 and CCP2, which share locations with RC1 and RC2, can be used for a PWM signal (see DC Motor tutorial). The pins AN0-AN7 are for analog I/O (see Photo resistor tutorial). TX and RX are for debugging I/O (see Output Messages to Computer tutorial). The remaining pins deal with power/ground, the clock signal, and programmer I/O. A PIC is made of several “ports.” Each port is designated with a letter, RB0-RB7 are a port. RC0-RC7 and RD0-RD7 are a port as well. RA0-RA5 and RE0-RE2 are also ports, but with fewer pins. Some of these pins have special purposes, but most can be used as basic input/output pins. For example, you can set pin RB0 to be either an input pin, or an output pin. As an input pin, the digital voltage on the pin can be read in. For example, if RB0 is connected to ground (0v), then you would read a digital 0. If RB0 was connected to power (5v), then you would read a digital 1. On the other hand, if you wanted to set RB0 as an output pin, you could choose to make RB0 either be 5v, or 0v.

Table 5.1 Register File Maps Used in Pic16f877

NAME	FUNCTION	ADDRESS RAM ADDRESS	BIT ADDRESABLE
STATUS	Status register	03H,83H,103H,1 83H	Yes
FSR	File select register	04H,84H,104H,1 84H	Yes
PORTA	I/O latch	05H	Yes
PORTB	I/O latch	06H	Yes
INTCON	Interrupt control register	0BH,8BH,10BH,1 8BH	Yes
PIR1	Peripheral interrupt	0CH	Yes
RCSTA	Receive status and control register	18H	Yes
TXREG	Transmit register	19H	Yes
RCREG	Receive register	1AH	Yes
OPTION_REG	Optional register	81H	Yes
TRISA	I/O register	85H	Yes
TRISB	I/O register	86H	Yes
TXSTA	Transmit status and control register	98H	Yes

5.9.4 Special Microcontroller Features

- Flash Memory: 14.3 Kbytes (8192 words)
- Data SRAM: 368 bytes
- Data EEPROM: 256 bytes
- Self-reprogrammable under software control
- In-Circuit Serial Programming via two pins (5V)
- Watchdog Timer with on-chip RC oscillator
- Programmable code protection
- Power-saving Sleep mode
- In-Circuit Debug via two pins
- 10-bit, 8-channel A/D Converter
- Brown-Out Reset
- Analog Comparator module

5.9.5 Peripheral Features

- 33 I/O pins; 5 I/O ports
- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
- Synchronous Serial Port with two modes:
 - SPI Master
 - I2C Master and Slave
- USART/SCI with 9-bit address detection
- Parallel Slave Port (PSP)
- Brown-out detection circuitry for Brown-Out Reset

5.10 REGISTER FILE

The term register file in PIC terminology used to denote the locations than an instruction can access via an address. The register file consists of two components, they are

1. General purpose register file
2. Special purpose register file

1. General Purpose Register File

The general-purpose register file is another name for the microcontrollers RAM. Data can be written to each 8-bit location, updated and retrieved any number of times. All control registers are coming under the general purpose register file.

2. Special Purpose Register File

The special purpose register file contains input and output ports as well as the control registers used to establish each bit of port as either an input or output. It contains registers that provide the data input and data output to the variety of resources on the chip, such as the timers, the serial ports and the analog-to-digital converter.

5.11 MEMORY ORGANIZATION

There are three memory blocks in each of the PIC16F87XA devices. The Program Memory and Data Memory have separate buses so that concurrent access.

5.11.1 Program Memory Organization

The PIC16F87XA devices have a 13-bit program counter capable of addressing an 8K word x 14 bit program memory space. The PIC16F876A/877A devices have 8K words x 14 bits of FLASH program memory, while PIC16F873A/874A devices have 4K words x 14 bits. Accessing a location above the physically implemented address will cause a wraparound. The RESET vector is at 0000h and the interrupt vector is at 0004h. The PIC16F87XA devices have a 13-bit program counter capable of addressing an 8K word x 14 bit program memory space.

The PIC16F876A/877A devices have 8K words x 14 bits of FLASH program memory, while PIC16F873A/874A devices have 4K words x 14 bits. Accessing a location above the

physically implemented address will cause a wraparound. The RESET vector is at 0000h and the interrupt vector is at 0004h

5.11.2 Data Memory Organization

The data memory is partitioned into multiple banks which contain the General Purpose Registers and the Special Function Registers. Bits RP1 (STATUS<6>) and RP0 (STATUS<5>) are the bank select bits.

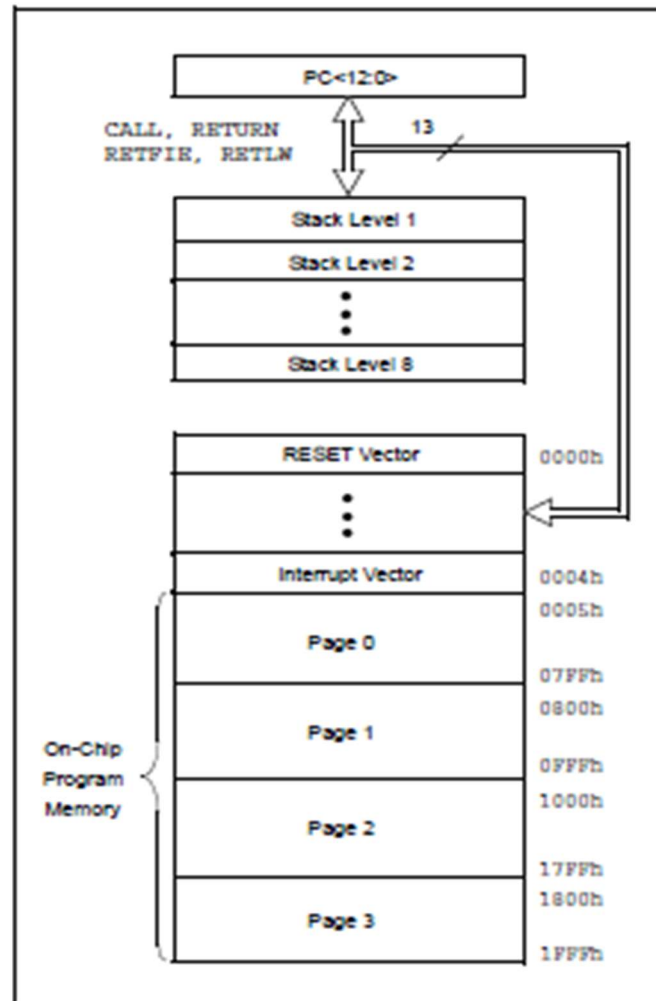


Figure: 5.4 Memory Organization Of Pic16f877a

Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. All implemented banks contain Special Function

Registers. Some frequently used Special Function Registers from one bank may be mirrored in another bank for code reduction and quicker access.

5.12 ADDRESSING MODES

5.12.1 Direct Addressing

Using this method we are accessing the registers directly by detecting location inside Data Memory from Opcode and by selecting the bank using bits RP1 and RP0 of the STATUS register.

5.12.2 Indirect Addressing

To implement indirect addressing, a File Select Register (FSR) and indirect register (INDF) are used. In addition, when using this method we choose bank using bit IRP of the STATUS register. Indirect addressing treated like a stack pointer, allowing much more efficient work with a number of variables. INDF register is not an actual register (it is a virtual register that is not found in any bank). To implement indirect addressing, a File Select Register (FSR) and indirect register (INDF) are used. In addition, when using this method we choose bank using bit IRP of the STATUS register. Indirect addressing treated like a stack pointer, allowing much more efficient work with a number of variables. INDF register is not an actual register (it is a virtual register that is not found in any bank). To implement indirect addressing, a File Select Register (FSR) and indirect register (INDF) are used. In addition, when using this method we choose bank using bit IRP of the STATUS register. Indirect addressing treated like a stack pointer, allowing much more efficient work with a number of variables. INDF register is not an actual register (it is a virtual register that is not found in any bank).

5.12.2.1 The following figure shows the two addressing methods:

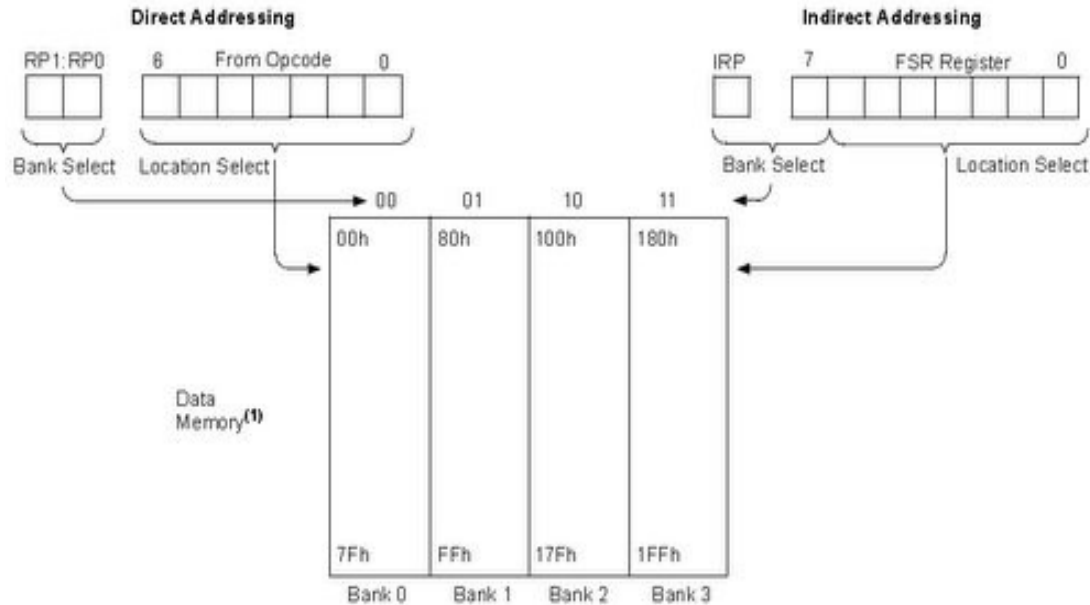


Fig: 5.5 Addressing Mode Diagram of PIC16F877A

5.13 PIC16F87XA DATA EEPROM

The data EEPROM and Flash program memory is readable and writable during normal operation (over the full VDD range). This memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers.

There are six SFRs used to read and write to this memory:

1. EECON1
2. EECON2
3. EEDATA
4. EEDATH
5. EEADR
6. EEADRH

A few important points about Data EEPROM memory:

- It lets you save data DURING programming
- The data is saved during the “burning” process
- You can read the data memory during the programming and use it
- The use is made possible with the help of SFR

At this point there is no need to learn how to use this memory with special registers, because there are functions (writing and reading) that are ready.

5.14 PIC TIMER:

Many times, we plan and build systems that perform various processes that depend on time. Simple example of this process is the digital wristwatch.

The role of this electronic system is to display time in a very precise manner and change the display every second (for seconds), every minute (for minutes) and so on.

To perform the steps we've listed, the system must use a timer, which needs to be very accurate in order to take necessary actions. The clock is actually a core of any electronic system.

In this PIC timer module tutorial we will study the existing PIC timer modules. The microcontroller PIC16F877 has 3 different timers:

- PIC Timer0
- PIC Timer1
- PIC Timer2

5.15 INPUT/OUTPUT PROGRAMMING IN PIC16F877A

General purpose I/O pins can be considered the simplest of peripherals. They allow the PIC microcontroller to monitor and control other devices. To add flexibility and functionality to a device, some pins are multiplexed with an alternate function(s). These functions depend on which peripheral features are on the device. In general, when a peripheral is functioning, that pin may not be used as a general purpose I/O pin. For most ports, the I/O pin's direction (input or output) is controlled by the data direction register, called the TRIS register. TRIS<x> controls the direction of PORT<x>. A '1' in the TRIS bit corresponds to that pin being an input, while a '0' corresponds to that pin being an output. An easy way to remember is that a '1' looks like I (input) and a '0' looks like an O output).

When peripheral functions are multiplexed onto general I/O pins, the functionality of the I/O pins may change to accommodate the requirements of the peripheral module. Examples of this are the Analog-to-Digital (A/D) converter and LCD driver modules, which force the I/O pin to the peripheral function when the device is reset. In the case of the A/D, this prevents the device from consuming excess current if any analog levels were on the A/D pins after a reset occurred.

With some peripherals, the TRIS bit is overridden while the peripheral is enabled. Therefore, read-modify-write instructions (BSF, BCF, XORWF) with TRIS as destination should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

PORT pins may be multiplexed with analog inputs and analog VREF input. The operation of each of these pins is selected, to be an analog input or digital I/O, by clearing/setting the control bits in the ADCON1 register (A/D Control Register1). When selected as an analog input, these pins will read as '0's.

The TRIS registers control the direction of the port pins, even when they are being used as analog inputs. The user must ensure the TRIS bits are maintained set when using the pins as analog inputs.

1) PORTA and the TRISA Register

- PORTA is a 6-bit wide bidirectional port (RA0 to RA5).
- TRISA register controls the direction of each pin.
- Used for both digital I/O and analog input (ADC).
- Example: TRISA = 0xFF; sets all PORTA pins as inputs.

2) PORTB and the TRISB Register

- PORTB is an 8-bit wide bidirectional port (RB0 to RB7).
- TRISB controls input/output direction.
- Pins RB4 to RB7 support **interrupt-on-change** functionality.
- Commonly used for digital input and output like LEDs and switches.

3) PORTC and the TRISC Register

- PORTC is an 8-bit wide port (RC0 to RC7).
- Supports peripheral functions like **UART, I²C, SPI, PWM**.
- TRISC sets the direction of each pin.
- Example: RC6 and RC7 are used for TX/RX in serial communication.

4) PORTD and the TRISD Register

- PORTD is an 8-bit port (RD0 to RD7), mostly used for **digital output**.
- Commonly used for interfacing **LED displays or relay modules**.
- TRISD = 0x00; sets all pins as output.

5) PORTE and the TRISE Register

- PORTE is a 3-bit wide port (RE0 to RE2).
- Can be configured as digital I/O or analog input.
- Used for **control signals or ADC channels** in many applications.
- TRISE sets pin direction.

5.16 LED PANEL



Fig: 5.6 LED

A **Traffic Indication LED Panel** is a type of electronic display used to convey traffic-related information to drivers at intersections or along roads. It uses **LED (Light Emitting Diode)** technology to display different colors, typically **red, green, and yellow** (or amber), to control the flow of traffic. In the context of the **Emergency Vehicle Detection System** that you've mentioned, the Traffic Indication LED Panels are specifically designed to indicate the status of the traffic at each path or lane. Here's how they work:

5.16.1 Function of Traffic Indication LED Panels in the System:

Red Light (Stop): When the system detects an emergency vehicle approaching a specific path, the corresponding indication panel turns **green** to allow the emergency vehicle to pass, while the other three indication panels turn **red** to stop other traffic and prevent congestion.

Green Light (Go): The indication panel for the path where the emergency vehicle is detected turns **green** to signal that this is the clear path for the emergency vehicle, allowing it to pass safely and without delay.

Yellow/Amber Light (Caution): Some systems may incorporate a yellow light to warn drivers about upcoming changes, like the approach of an emergency vehicle. This can act as a transitional state before switching from red to green or vice versa.

5.16.2 Features of the Traffic Indication LED Panels:

Energy Efficiency: LED lights are energy-efficient, long-lasting, and more durable than traditional light bulbs.

Clear Visibility: LEDs are bright and visible from a distance, making them easy for drivers to see and follow, even in poor weather or at night.

Quick Response: LED panels can switch between colors very quickly, providing real-time updates to drivers.

Customization: These panels can be customized to display specific messages or warnings (such as "Emergency Vehicle Approaching").

5.16.3 Application in Emergency Vehicle Detection System:

Each path or lane in an intersection is equipped with a **Traffic Indication LED Panel**. The **AI-based vehicle detection system** detects the presence of an emergency vehicle and activates the corresponding LED panel to turn **green**. The other three panels will turn **red** to block the other traffic and give priority to the emergency vehicle. Once the emergency vehicle passes and is no longer detected by the AI system, the system will reset the traffic control back to normal, and all panels will return to their regular function.

CHAPTER 6

SOFTWARE DESCRIPTION

SOFTWARE SPECIFICATION

- OPERATING SYSTEM : WINDOWS 7/10/11
- LANGUAGE USED : PYTHON

6.1 MPLAB IDE SOFTWARE

MPLAB is a proprietary freeware integrated development environment for the development of embedded applications on PIC microcontrollers, and is developed by Microchip Technology.

MPLAB X is the latest edition of MPLAB and is developed on the NetBeans platform. MPLAB and MPLAB X support project management, code editing, debugging and programming of Microchip 8-bit, 16-bit and 32-bit PIC microcontrollers.

MPLAB is designed to work with MPLAB-certified devices such as the MPLAB ICD 3 and MPLAB REAL ICE, for programming and debugging PIC microcontrollers using a personal computer. PICK it programmers are also supported by MPLAB.

MPLAB 8.X is the last version of the legacy MPLAB IDE technology, custom built by Microchip Technology in Microsoft Visual C++. MPLAB supports project management, editing, debugging and programming of Microchip 8-bit, 16-bit and 32-bit Pic microcontrollers. MPLAB only works on Microsoft Windows. MPLAB is still available from Microchip's archives but is not recommended for new projects.

MPLAB supports the following compilers:

MPLAB MPASM Assembler

MPLAB ASM30 Assembler

MPLAB C Compiler for PIC18

MPLAB C Compiler for PIC24 and dsPIC DSCs

MPLAB C Compiler for PIC32

HI-TECH C

MPLAB X is the latest version of the MPLAB IDE built by Microchip Technology, and is based on the open-source NetBeans platform. MPLAB X supports editing, debugging and programming of Microchip 8-bit, 16-bit and 32-bit PIC microcontrollers.

MPLAB X is the first version of the IDE to include cross-platform support for Mac OS X and Linux operating systems, in addition to Microsoft Windows.

MPLAB X supports the following compilers:

- MPLAB XC8 — C compiler for 8-bit PIC devices
- MPLAB XC16 — C compiler for 16-bit PIC devices
- MPLAB XC32 — C/C++ compiler for 32-bit PIC devices
- HI-TECH C — C compiler for 8-bit PIC devices
- SDCC — open-source C compiler
- HI-TECH C compiler for PIC10/12/16 MCUs (PRO)

This compiler has been discontinued and is no longer supported. This compiler has been replaced by the MPLAB® XC8 PRO (SW006021-2).

HI-TECH C Compiler for PIC10/12/16 MCUs - PRO fully implements the optimizations of Omniscent Code Generation™ - a whole-program compilation technology - to provide denser code and better performance on PIC MCUs. This ANSI C compiler integrates into Microchips MPLAB(R) IDE and is compatible with Microchip debuggers and emulators.

6.1.1 MPLAB IDE

- Free integrated development environment (IDE) from Microchip to implement code for PICs
- Latest version 8.60(recommended), In Lab 7.6
- IDE and documentation (user guide) can be downloaded from the Microchip website
- To open MPLAB IDE
- **Start→All Programs→Microchip→MPLAB IDE v8.60→MPLAB IDE**

6.1.2 PROJECT CREATION

A project must be created for implementation

- Specify your device
- Create and edit your files
- Compile and link your project
- Program the device

To create a project

- Project→Project Wizard...
- Begin project and specify device
- Select Microchip MPASM Tool suite
- Create New Project File
- Add project files

- Finish project creation and return to IDE

6.1.3 Updating Source Code

- Modify .asm files
- Under 'Source Files' folder in project window
- Open and edit files to implement new functionality
- Additional files can be created and added

6.1.4 Building Projects

- Project → Build All (or Ctrl+F10)
- Output window indicates success or failure
- HEX file generated when project is built

6.1.5 Programming the Pic

Directly Downloading Hex files to the PIC Memory

- Used to transfer HEX file to the PIC and begin program execution
- Two methods (HEX file must be generated)
 - MPLAB IDE
 - PICkit3

6.2 PYTHON

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- Web development (server-side),
- Software development,
- Mathematics,
- System scripting.

6.3 PYTHON NUMPY

Our Python NumPy Tutorial provides the basic and advanced concepts of the NumPy. Our NumPy tutorial is designed for beginners and professionals. NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements.



It is an extension module of Python which is mostly written in C. It provides various functions which are capable of performing the numeric computations with a high speed. NumPy provides various powerful data structures, implementing multi-dimensional arrays and matrices. These data structures are used for the optimal computations regarding arrays and matrices. NumPy provides a convenient and efficient way to handle the vast amount of

data. NumPy is also very convenient with Matrix multiplication and data reshaping. NumPy is fast which makes it reasonable to work with a large set of data.

There are the following advantages of using NumPy for data analysis.

1. NumPy performs array-oriented computing.
2. It efficiently implements the multidimensional arrays.
3. It performs scientific computations.
4. It is capable of performing Fourier Transform and reshaping the data stored in multidimensional arrays.
5. NumPy provides the in-built functions for linear algebra and random number generation.

Nowadays, NumPy in combination with SciPy and Matplotlib is used as the replacement to MATLAB as Python is more complete and easier programming language than MATLAB. Prerequisite Before learning Python Numpy, you must have the basic knowledge of Python concepts.

6.4 OPEN CV

Open CV tutorial provides basic and advanced concepts of OpenCV. Our Open CV tutorial is designed for beginners and professionals. OpenCV is an open-source library for the computer vision. It provides the facility to the machine to recognize the faces or objects. In this tutorial we will learn the concept of OpenCV using the Python programming language. Our OpenCV tutorial includes all topics of Read and Save Image, Canny Edge Detection, Template matching, Blob Detection, Contour, Mouse Event, Gaussian blur and so on.

6.4.1 What is OpenCV?

OpenCV is a Python open-source library, which is used for computer vision in Artificial intelligence, Machine Learning, face recognition, etc.



In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and videos.

The purpose of computer vision is to understand the content of the images. It extracts the description from the pictures, which may be an object, a text description, and three-dimension model, and so on. For example, cars can be facilitated with computer vision, which will be able to identify and different objects around the road, such as traffic lights, pedestrians, traffic signs, and so on, and acts accordingly. Computer vision allows the computer to perform the same kind of tasks as humans with the same efficiency. There are a two main task which are defined below:

- **Object Classification** - In the object classification, we train a model on a dataset of particular objects, and the model classifies new objects as belonging to one or more of your training categories.

- **Object Identification** - In the object identification, our model will identify a particular instance of an object - for example, parsing two faces in an image and tagging one as Virat Kohli and other one as Rohit Sharma.

How does computer recognize the image?

Human eyes provide lots of information based on what they see. Machines are facilitated with seeing everything, convert the vision into numbers and store in the memory. Here the question arises how computer convert images into numbers. So the answer is that the pixel value is used to convert images into numbers. A pixel is the smallest unit of a digital image or graphics that can be displayed and represented on a digital display device.

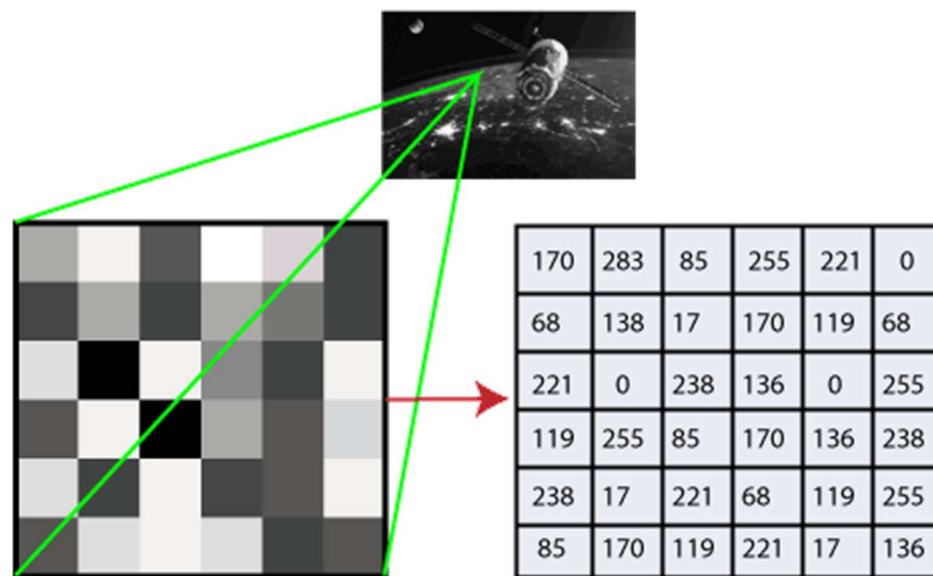


Fig :6.1 Pixel Recognition

The picture intensity at the particular location is represented by the numbers. In the above image, we have shown the pixel values for a grayscale image consist of only one value, the intensity of the black color at that location.

There are two common ways to identify the images:

1. Grayscale

Grayscale images are those images which contain only two colors black and white. The contrast measurement of intensity is black treated as the weakest intensity, and white as the strongest intensity. When we use the grayscale image, the computer assigns each pixel value based on its level of darkness.

2. RGB

An RGB is a combination of the red, green, blue color which together makes a new color. The computer retrieves that value from each pixel and puts the results in an array to be interpreted.

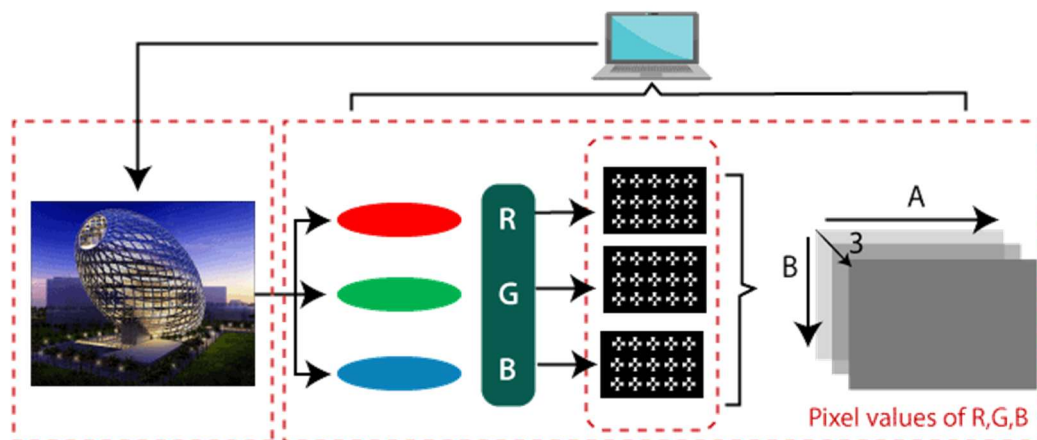


Fig : 6.2 RGB

6.4.3 Why OpenCV is used for Computer Vision?

- OpenCV is available for free of cost.
- Since the OpenCV library is written in C/C++, so it is quit fast. Now it can be used with Python.
- It Require less RAM to usage, it maybe of 60-70 MB.

- Computer Vision is portable as OpenCV and can run on any device that can run on C.

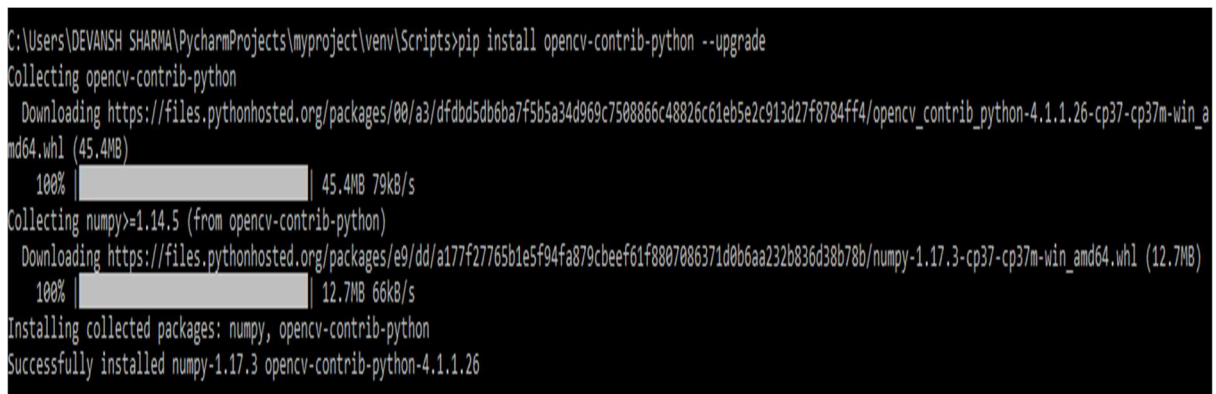
6.4.4 Installation of the OpenCV

The first step is to download the latest Anaconda graphic installer for Windows from its official site. Choose your bit graphical installer. You are suggested to install 3.7 working with Python 3.

Install OpenCV in the Windows via pip

OpenCV is a Python library so it is necessary to install Python in the system and install OpenCV using pip command:

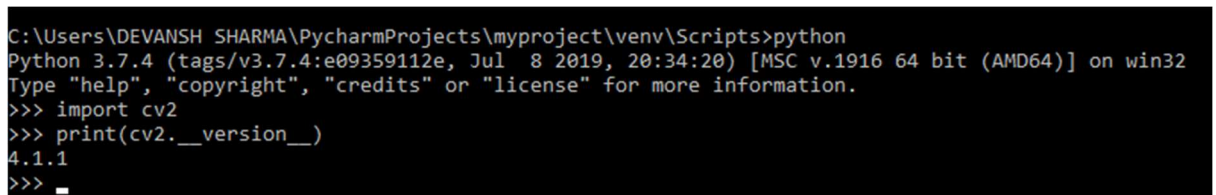
1.pip install opencv-python



```
C:\Users\DEVANSH SHARMA\PycharmProjects\myproject\venv\Scripts>pip install opencv-contrib-python --upgrade
Collecting opencv-contrib-python
  Downloading https://files.pythonhosted.org/packages/00/a3/dfdbd5db6ba7f5b5a34d969c7508866c48826c61eb5e2c913d27f8784ff4/opencv_contrib_python-4.1.1.26-cp37-cp37m-win_a
md64.whl (45.4MB)
    100% |#####| 45.4MB 79kB/s
Collecting numpy>=1.14.5 (from opencv-contrib-python)
  Downloading https://files.pythonhosted.org/packages/e9/dd/a177f27765b1e5f94fa879cbeef61f8807086371d0b6aa232b836d38b78b/numpy-1.17.3-cp37-cp37m-win_amd64.whl (12.7MB)
    100% |#####| 12.7MB 66kB/s
Installing collected packages: numpy, opencv-contrib-python
Successfully installed numpy-1.17.3 opencv-contrib-python-4.1.1.26
```

Fig: 6.3 Pip Installation

2.Open the command prompt and type the following code to check if the OpenCV is installed or not.



```
C:\Users\DEVANSH SHARMA\PycharmProjects\myproject\venv\Scripts>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> print(cv2.__version__)
4.1.1
>>> _
```

Fig: 6.4 Command prompt

6.5 SCIKIT-LEARN:



Scikit-learn is probably the most useful library for machine learning in Python. It is on NumPy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. Scikit-learn is an open source Python library that has powerful tools for data analysis and data mining. It's available under the BSD license and is built on the following machine learning libraries: ... SciPy, an ecosystem consisting of various libraries for completing technical computing tasks. Scikit-Learn has a good number of ML algorithms which can be readily deployed in our models. It doesn't require the programmer to be a pro in machine learning concepts. It just needs us to specify what needs to be done. Not to mention, it is a high level library with brutal abstractions.

6.6 PANDAS:



Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.[2] The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

6.6.1 Library feature:

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

6.7 WEB CAMERA:

- A **webcam** is a hardware **camera** and input **device** that connects to a computer and the **Internet** and captures either still pictures or motion video of a driver drowsiness.
- A **webcam** is a video **camera** that feeds or streams an image or video in real time to or through a computer to a computer network, such as the Internet. Webcams are typically small **cameras** that sit on a desk, attach to a user's monitor, or are built into the hardware
- Webcams are usually cheaper than a standard video **camera** and allow for face-to-face communication online, making it easy to illustrate things visually to the person you are talking to. This makes the **webcam** a very versatile device for home or office use

6.8 TENSORFLOW- INTRODUCTION

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions. The official website of TensorFlow is mentioned below:

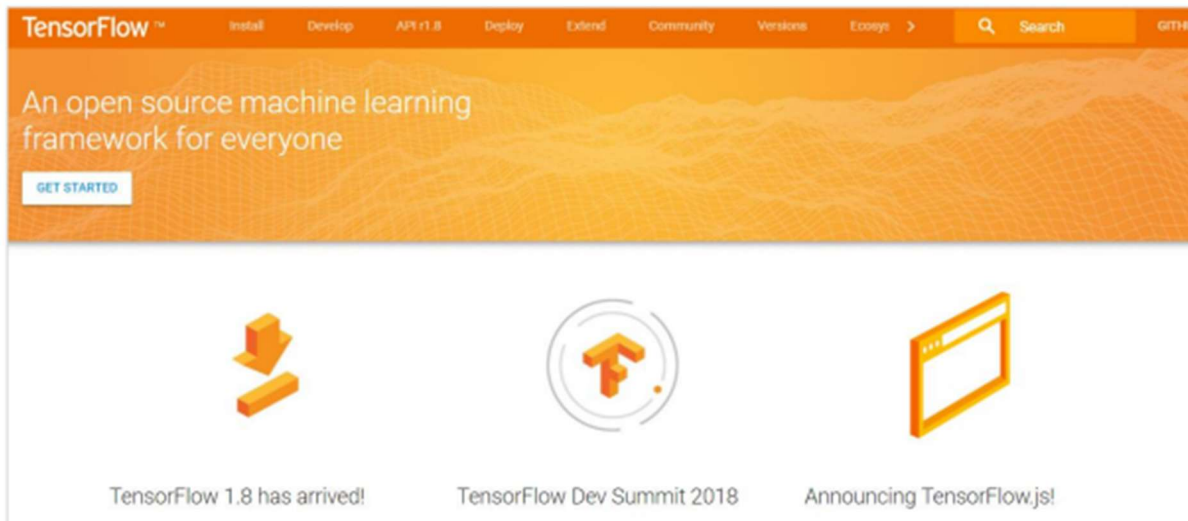


Fig: 6.4 TensorFlow

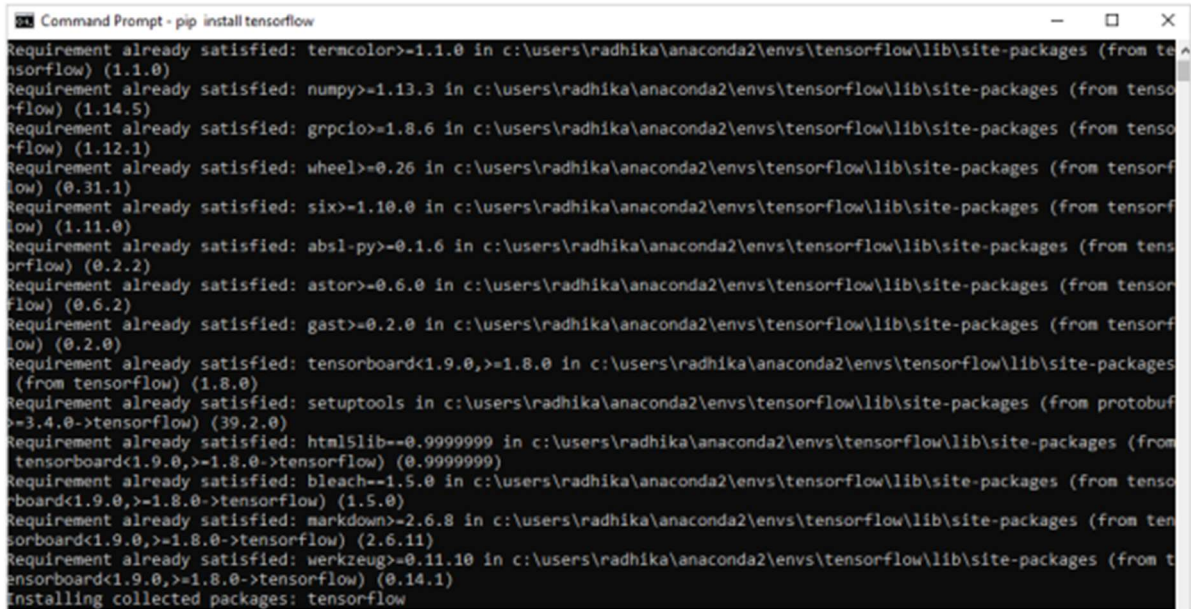
Let us now consider the following important features of TensorFlow:

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

TensorFlow — Installation

To install TensorFlow, it is important to have “Python” installed in your system. Python version 3.4+ is considered the best to start with TensorFlow installation. Consider the following steps to install TensorFlow in Windows operating system.

`pip install tensorflow`



```
Command Prompt - pip install tensorflow
Requirement already satisfied: termcolor>=1.1.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.14.5)
Requirement already satisfied: grpcio>=1.8.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: wheel>=0.26 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.31.1)
Requirement already satisfied: six>=1.10.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.11.0)
Requirement already satisfied: absl-py>=0.1.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.2.2)
Requirement already satisfied: astor>=0.6.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.6.2)
Requirement already satisfied: gast>=0.2.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: tensorboard<1.9.0,>=1.8.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.8.0)
Requirement already satisfied: setuptools in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (39.2.0)
Requirement already satisfied: html5lib==0.9999999 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.9999999)
Requirement already satisfied: bleach==1.5.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.5.0)
Requirement already satisfied: markdown==2.6.8 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (2.6.11)
Requirement already satisfied: werkzeug>=0.11.10 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.14.1)
Installing collected packages: tensorflow
```

Fig: 6.5 TensorFlow Installation

Following are the two **TensorFlow — Convolutional Neural Networks**

After understanding machine-learning concepts, we can now shift our focus to deep learning concepts. Deep learning is a division of machine learning and is considered as a crucial step taken by researchers in recent decades. The examples of deep learning implementation include applications like image recognition and speech recognition.

Important types of deep neural networks:

- Convolutional Neural Networks
- Recurrent Neural Networks In this chapter, we will focus on the CNN, Convolutional Neural Networks

6.9 CONVOLUTIONAL NEURAL NETWORK (CNN)

Convolutional neural network is one of the most popular ANN. It is widely used in the fields of image and video recognition. It is based on the concept of convolution, a mathematical concept. It is almost similar to multi-layer perceptron except it contains series of convolution layer and pooling layer before the fully connected hidden neuron layer.

It has three important layers:

- **Convolution layer:** It is the primary building block and perform computational tasks based on convolution function.
- **Pooling layer:** It is arranged next to convolution layer and is used to reduce the size of inputs by removing unnecessary information so computation can be performed faster.
- **Fully connected layer:** It is arranged to next to series of convolution and pooling layer and classify input into various categories.
- 2 series of Convolution and pooling layer is used and it receives and process the input (e.g. image).
- A single fully connected layer is used and it is used to output the data (e.g. classification of image)

Convolutional Neural networks are designed to process data through multiple layers of arrays. This type of neural networks is used in applications like image recognition or face recognition. The primary difference between CNN and any other ordinary neural network is

that CNN takes input as a two-dimensional array and operates directly on the images rather than focusing on feature extraction which other neural networks focus on. The dominant approach of CNN includes solutions for problems of recognition. Top companies like Google and Facebook have invested in research and development towards recognition projects to get activities done with greater speed.

6.9.1 Layers in CNN

There are five different layers in CNN

- Input layer
- Convo layer (Convo + ReLU)
- Pooling layer
- Fully connected (FC) layer
- Softmax/logistic layer
- Output layer

Input Layer

Input layer in CNN should contain image data. Image data is represented by three dimensional matrix as we saw earlier. You need to reshape it into a single column.

Suppose you have image of dimension $28 \times 28 = 784$, you need to convert it into 784×1 before feeding into input. If you have “m” training examples then dimension of input will be $(784, m)$.

Convo Layer

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter)

and the filter. Result of the operation is single integer of the output volume. Then we slide the filter over the next receptive field of the same input image by a Stride and do the same operation again. We will repeat the same process again and again until we go through the whole image. The output will be the input for the next layer. Convo layer also contains ReLU activation to make all negative value to zero.

Pooling Layer

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layer. If we apply FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive and we don't want it. So, the max pooling is only way to reduce the spatial volume of input image. In the above example, we have applied max pooling in single depth slice with Stride of 2. You can observe the 4 x 4 dimension input is reduce to 2 x 2 dimension.

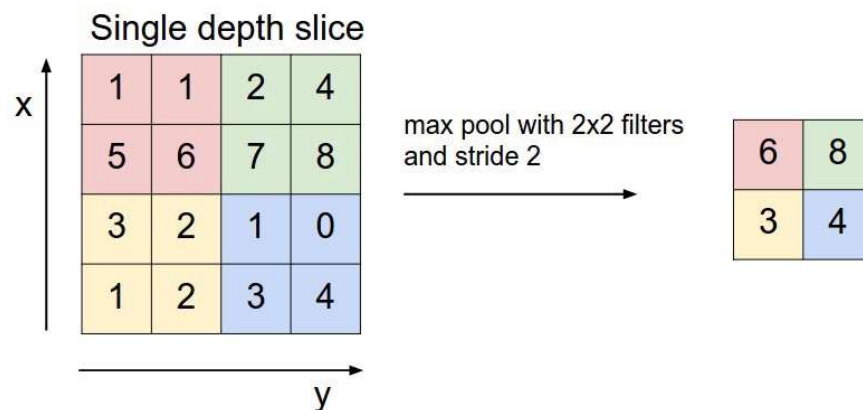


Fig: 6.6 Pooling Layer

There is no parameter in pooling layer but it has two hyper parameters — Filter(F) and Stride(S).

Fully Connected Layer (FC)

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

Softmax / Logistic Layer

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

Output Layer

Output layer contains the label which is in the form of one-hot encoded.

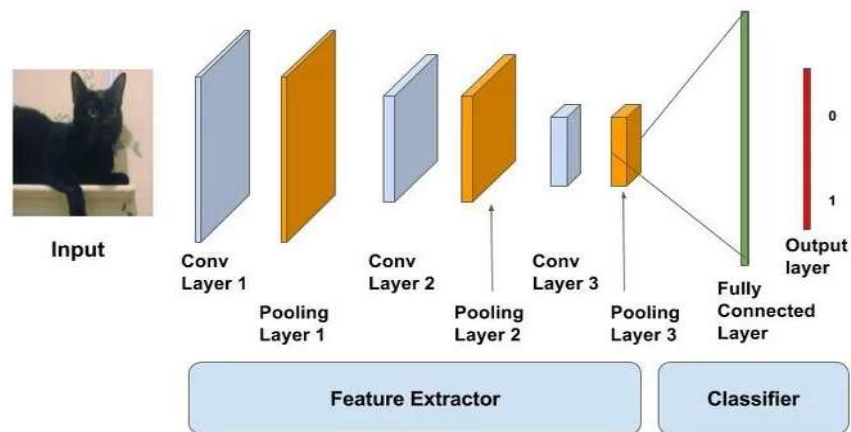


Fig: 6.7 Output layer

CHAPTER 7

METHODOLOGY AND OVERVIEW

7.1 METHODOLOGY

Convolutional neural network basics The main building blocks of a deep convolutional neural network (CNN) are the convolutional layer, commonly called CONV2D, the pooling layer, the rectified linear unit (ReLU) layer, and a series of fully connected layers. A deep CNN basically consists of two subnetwork, a series of 2D convolutional layers and a classical but deep neural network.

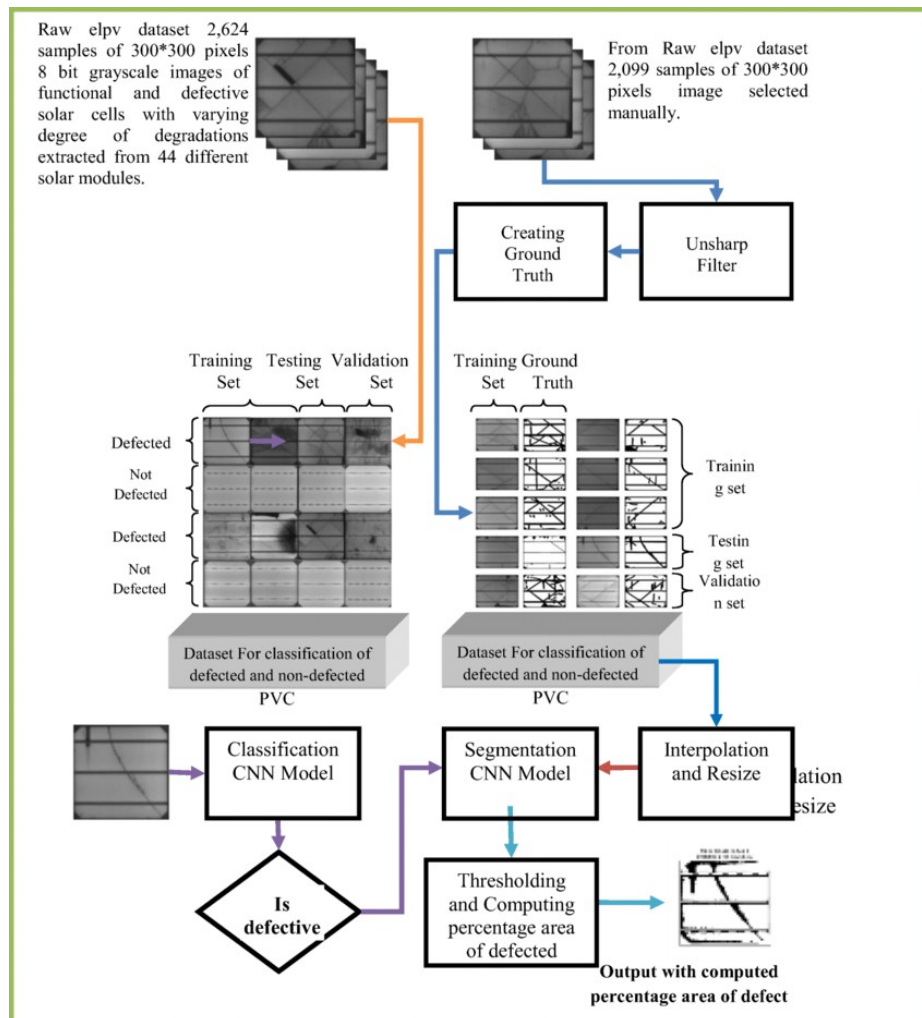


Fig: 7.1 METHODOLOGY

7.2 PYTHON PILLOW — OVERVIEW

In today's digital world, we come across lots of digital images. In case, we are working with Python programming language, it provides lot of image processing libraries to add image processing capabilities to digital images.

Some of the most common image processing libraries are: OpenCV, Python Imaging Library (PIL), Scikit-image, Pillow. However, in this tutorial, we are only focusing on Pillow module and will try to explore various capabilities of this module.

Pillow is built on top of PIL (Python Image Library). PIL is one of the important modules for image processing in Python. However, the PIL module is not supported since 2011 and doesn't support python 3.

Pillow module gives more functionalities, runs on all major operating system and support for python 3. It supports wide variety of images such as “jpeg”, “png”, “bmp”, “gif”, “ppm”, “tiff”. You can do almost anything on digital images using pillow module. Apart from basic image processing functionality, including point operations, filtering images using built-in convolution kernels, and color space conversions.

Image Archives

The Python Imaging Library is best suited for image archival and batch processing applications. Python pillow package can be used for creating thumbnails, converting from one format to another and print images, etc.

Image Display

You can display images using Tk PhotoImage, BitmapImage and Windows DIB interface, which can be used with PythonWin and other Windows-based toolkits and many other Graphical User Interface (GUI) toolkits.

For debugging purposes, there is a `show ()` method to save the image to disk which calls the external display utility.

Image Processing

The Pillow library contains all the basic image processing functionality. You can do image resizing, rotation and transformation.

Pillow module allows you to pull some statistics data out of image using histogram method, which later can be used for statistical analysis and automatic contrast enhancement.

7.3 PYTHON PILLOW — ENVIRONMENT SETUP

This chapter discusses how to install pillow package in your computer.

Installing pillow package is very easy, especially if you're installing it using pip.

Installing Pillow using pip

To install pillow using pip, just run the below command in your command prompt:

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

```
python -m pip install pillow
```

In case, if pip and pillow are already installed in your computer, above commands will simply mention the 'requirement already satisfied' as shown below:

```
C:\Users\yadur>python -m pip install pip
Requirement already satisfied: pip in c:\python381\lib\site-packages (19.3.1)

C:\Users\yadur>python -m pip install pillow
Requirement already satisfied: pillow in c:\python381\lib\site-packages (7.0.0)
```

Fig: 7.2 Installing pillow using pip

7.4 PYTHON PILLOW — USING IMAGE MODULE

To display the image, pillow library is using an image class within it. The image module inside pillow package contains some important inbuilt functions like, load images or create new images, etc.

7.5 KERAS INTRODUCTION

Deep learning is one of the major subfield of machine learning framework. Machine learning is the study of design of algorithms, inspired from the model of human brain. Deep learning is becoming more popular in data science fields like robotics, artificial intelligence (AI), audio & video recognition and image recognition. Artificial neural network is the core of deep learning methodologies. Deep learning is supported by various libraries such as Theano, TensorFlow, Caffe, Mxnet etc., Keras is one of the most powerful and easy to use python library, which is built on top of popular deep learning libraries like TensorFlow, Theano, etc., for creating deep learning models.

7.5.2 Features of Keras

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

7.5.3 Benefits of Keras

Keras is highly powerful and dynamic framework and comes up with the following advantages:

- Larger community support.
- Easy to test.
- Keras neural networks are written in Python which makes things simpler.
- Keras supports both convolution and recurrent networks.
- Deep learning models are discrete components, so that, you can combine into many ways.

7.6 ARTIFICIAL NEURAL NETWORKS

The most popular and primary approach of deep learning is using “Artificial neural network” (ANN). They are inspired from the model of human brain, which is the most complex organ of our body.

The human brain is made up of more than 90 billion tiny cells called “Neurons”. Neurons are inter-connected through nerve fiber called “axons” and “Dendrites”.

The main role of axon is to transmit information from one neuron to another to which it is connected.

Similarly, the main role of dendrites is to receive the information being transmitted by the axons of another neuron to which it is connected.

Each neuron processes a small information and then passes the result to another neuron and this process continues.

This is the basic method used by our human brain to process huge amount of information like speech, visual, etc., and extract useful information from it.

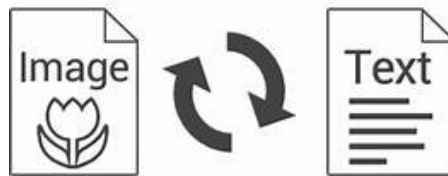
7.6.1 Single Layer perceptron

- Multiple input along with weight represents dendrites.
- Sum of input along with activation function represents neurons. Sum actually means computed value of all inputs and activation function represent a function, which modify the Sum value into 0, 1 or 0 to 1.
- Actual output represent axon and the output will be received by neuron in next layer.

7.6.2 Multi-Layer Perceptron

Multi-Layer perceptron is the simplest form of ANN. It consists of a single input layer, one or more hidden layer and finally an output layer. A layer consists of a collection of perceptron. Input layer is basically one or more features of the input data. Every hidden layer consists of one or more neurons and process certain aspect of the feature and send the processed information into the next hidden layer. The output layer process receives the data from last hidden layer and finally output the result.

7.7 OPTICAL CHARACTER RECOGNITION (OCR)



The ability of machines to use a camera to look at the real world and interpret data from it would have a greater influence on its applications. Be it a simple food delivery Robot like the Starship Robots or an advanced self driving car like Tesla, they are rely on information

obtained from their highly sophisticated cameras to take decisions. In this tutorial we will learn how details are identified from images by reading the characters present on it. This is called Optical Character Recognition (OCR). This opens door for many applications like to automatically read the information from a business card, recognize a shop from its name board or recognize sign boards on road and much more. Some of us might have already experienced these features through Google Lens, so today we will build something similar using an Optical Character Recognition (OCR) Tool from Google Tesseract-OCR Engine along with python and OpenCV to identify characters from pictures..Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

7.7.1 Implementation OCR

Optical character recognition, or OCR, is a method of converting a scanned image into text. When a page is scanned, it is typically stored as a bit-mapped file in TIF format. When the image is displayed on the screen, we can read it. ... The computer does not recognize any "words" on the image. OCR or Optical Character Recognition is a software application included with certain HP scanners. Traditionally, documents scanned into a computer are saved as PDFs and can be read only on a computer. OCR allows a user to scan documents and save them to a computer, but also to be able to edit the documents.

7.8 PYTESSERACT:

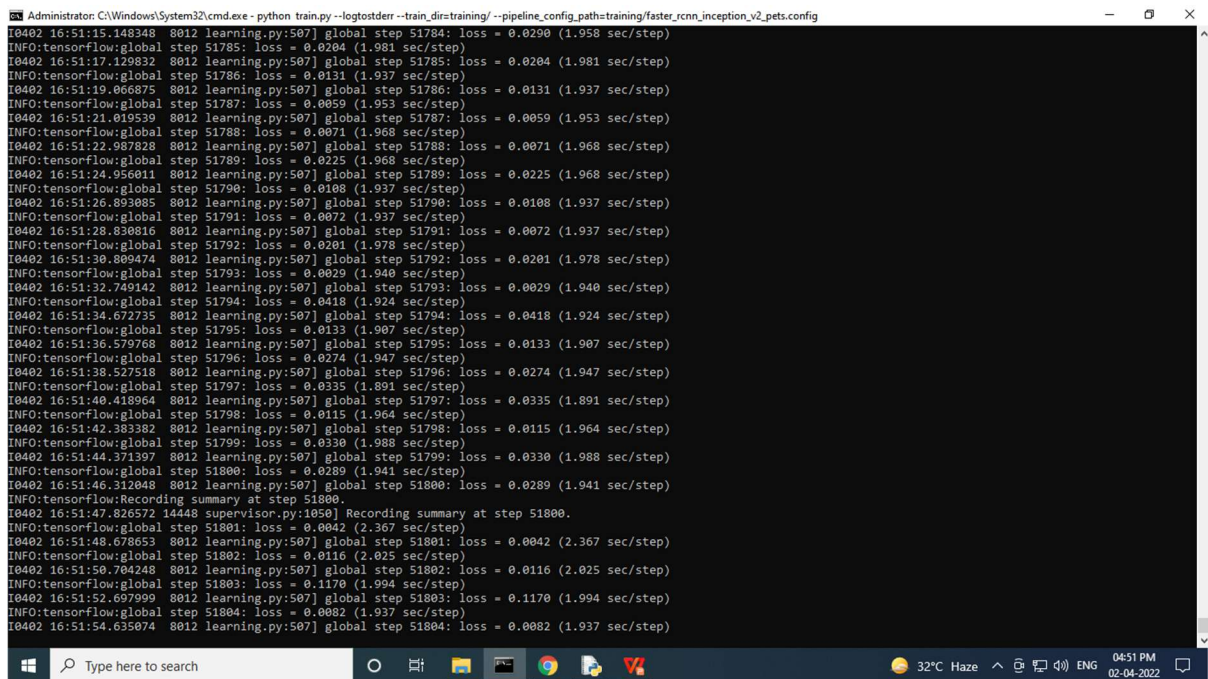
Pytesseract allows us to configure the Tesseract OCR engine by setting the flags which changes the way in which the image is searched for characters. The three main flags used in configuring a Tesseract OCR is language (-l), OCR Engine Mode (--oem) and Page

Segmentation Mode (- -psm). Along with the default English language, Tesseract supports many other languages including Hindi, Turkish, French etc.

7.9 TRAINING THE MODEL

Because the data required to train a CNN is very large, it is often desirable to train the model in batches. Loading all the training data into memory is not always possible because you need enough memory to handle it and the features too. So we decided to load all the images into a kaggle dataset using faster rcnn model.

Once I had the training data in a kaggle dataset, I trained the model.



```
Administrator: C:\Windows\System32\cmd.exe - python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/faster_rcnn_inception_v2_pets.config
[0402 16:51:15.148348 8012 learning.py:507] global step 51784: loss = 0.0290 (1.958 sec/step)
INFO:tensorflow:global step 51785: loss = 0.0204 (1.981 sec/step)
[0402 16:51:17.129832 8012 learning.py:507] global step 51785: loss = 0.0204 (1.981 sec/step)
INFO:tensorflow:global step 51786: loss = 0.0131 (1.937 sec/step)
[0402 16:51:19.066875 8012 learning.py:507] global step 51786: loss = 0.0131 (1.937 sec/step)
INFO:tensorflow:global step 51787: loss = 0.0059 (1.953 sec/step)
[0402 16:51:21.019539 8012 learning.py:507] global step 51787: loss = 0.0059 (1.953 sec/step)
INFO:tensorflow:global step 51788: loss = 0.0071 (1.968 sec/step)
[0402 16:51:22.987828 8012 learning.py:507] global step 51788: loss = 0.0071 (1.968 sec/step)
INFO:tensorflow:global step 51789: loss = 0.0225 (1.968 sec/step)
[0402 16:51:24.956011 8012 learning.py:507] global step 51789: loss = 0.0225 (1.968 sec/step)
INFO:tensorflow:global step 51790: loss = 0.0108 (1.937 sec/step)
[0402 16:51:26.893085 8012 learning.py:507] global step 51790: loss = 0.0108 (1.937 sec/step)
INFO:tensorflow:global step 51791: loss = 0.0072 (1.937 sec/step)
[0402 16:51:28.830816 8012 learning.py:507] global step 51791: loss = 0.0072 (1.937 sec/step)
INFO:tensorflow:global step 51792: loss = 0.0201 (1.978 sec/step)
[0402 16:51:30.809474 8012 learning.py:507] global step 51792: loss = 0.0201 (1.978 sec/step)
INFO:tensorflow:global step 51793: loss = 0.0029 (1.940 sec/step)
[0402 16:51:32.749142 8012 learning.py:507] global step 51793: loss = 0.0029 (1.940 sec/step)
INFO:tensorflow:global step 51794: loss = 0.0418 (1.924 sec/step)
[0402 16:51:34.672735 8012 learning.py:507] global step 51794: loss = 0.0418 (1.924 sec/step)
INFO:tensorflow:global step 51795: loss = 0.0133 (1.907 sec/step)
[0402 16:51:36.579768 8012 learning.py:507] global step 51795: loss = 0.0133 (1.907 sec/step)
INFO:tensorflow:global step 51796: loss = 0.0274 (1.947 sec/step)
[0402 16:51:38.527518 8012 learning.py:507] global step 51796: loss = 0.0274 (1.947 sec/step)
INFO:tensorflow:global step 51797: loss = 0.0335 (1.891 sec/step)
[0402 16:51:40.418964 8012 learning.py:507] global step 51797: loss = 0.0335 (1.891 sec/step)
INFO:tensorflow:global step 51798: loss = 0.0115 (1.964 sec/step)
[0402 16:51:42.383382 8012 learning.py:507] global step 51798: loss = 0.0115 (1.964 sec/step)
INFO:tensorflow:global step 51799: loss = 0.0330 (1.988 sec/step)
[0402 16:51:44.371397 8012 learning.py:507] global step 51799: loss = 0.0330 (1.988 sec/step)
INFO:tensorflow:global step 51800: loss = 0.0289 (1.941 sec/step)
[0402 16:51:46.312048 8012 learning.py:507] global step 51800: loss = 0.0289 (1.941 sec/step)
INFO:tensorflow:Recording summary at step 51800.
[0402 16:51:47.826572 14448 supervisor.py:1050] Recording summary at step 51800.
INFO:tensorflow:global step 51801: loss = 0.0042 (2.367 sec/step)
[0402 16:51:48.678653 8012 learning.py:507] global step 51801: loss = 0.0042 (2.367 sec/step)
INFO:tensorflow:global step 51802: loss = 0.0116 (2.025 sec/step)
[0402 16:51:50.704248 8012 learning.py:507] global step 51802: loss = 0.0116 (2.025 sec/step)
INFO:tensorflow:global step 51803: loss = 0.1170 (1.994 sec/step)
[0402 16:51:52.697999 8012 learning.py:507] global step 51803: loss = 0.1170 (1.994 sec/step)
INFO:tensorflow:global step 51804: loss = 0.0082 (1.937 sec/step)
[0402 16:51:54.635074 8012 learning.py:507] global step 51804: loss = 0.0082 (1.937 sec/step)
```

Fig: 7.3 TRAINING THE MODEL

7.10 APPLICATION

- **Helmet Violation Detection:** Automatically identifies and reports two-wheeler riders without helmets.
- **Stolen Vehicle Detection:** Recognizes number plates and flags vehicles listed in stolen databases.
- **Emergency Vehicle Prioritization:** Detects ambulances or fire trucks and clears their path by adjusting signals.
- **Dynamic Traffic Signal Control:** Adjusts red/green light durations based on real-time traffic congestion.
- **Smart Surveillance for Law Enforcement:** Supports automated monitoring and evidence collection for traffic violations

7.11 ADVANTAGES

- **Real-time Monitoring:** Provides instant analysis and response to live traffic conditions.
- **Improved Road Safety:** Encourages helmet use and assists in catching traffic violators.
- **Reduced Congestion:** Dynamically manages signal timings based on actual traffic flow.
- **Emergency Efficiency:** Enhances emergency response by clearing paths for ambulances and fire trucks.
- **Automation and Scalability:** Minimizes human error and can be deployed across large urban areas.

CHAPTER 8

PROTOTYPE

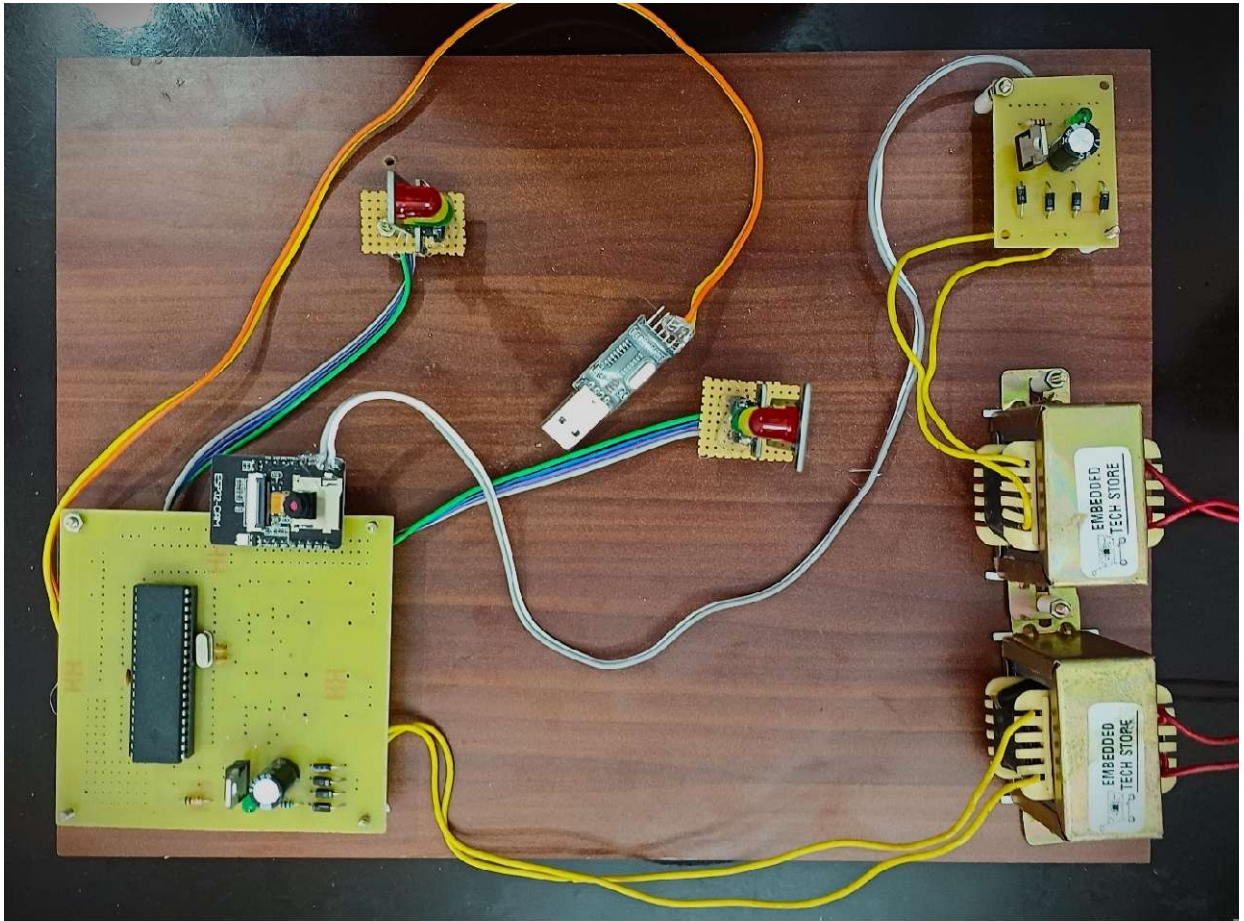


Fig: 8.1 PROTOTYPE

CHAPTER 9

CONCLUSION

The implementation of an intelligent traffic management system using ESP32-CAM, machine learning, and OCR has proven to be an effective approach for enhancing urban mobility, safety, and enforcement. The system successfully automates helmet violation detection, number plate recognition, and dynamic signal control based on real-time traffic and emergency scenarios. Through practical testing, it demonstrated high accuracy and responsiveness, supporting its feasibility for smart city integration.

By reducing reliance on manual intervention and enabling rapid decision-making, the proposed system offers a scalable and cost-effective solution to modern traffic challenges. While further improvements can be made in low-light recognition and audio-based emergency detection, the current system lays a strong foundation for future enhancements. Overall, this project contributes significantly toward building safer, smarter, and more efficient urban traffic environments.

REFERENCES

- 1) Guo, N., Zhang, X., Zou, Y., 2022. Real-time predictive control of path following to stabilize autonomous electric vehicles under extreme drive conditions. *Automot Innov*, 5, 453–470.
- 2) He, Y., Liu, Y., Yang, L., Qu, X. 2023. Deep adaptive control: Deep reinforcement learning-based adaptive vehicle trajectory control algorithms for different risk levels. *IEEE Trans Intell Veh*, 9, 1654–1666.
- 3) Jaswanth, M., Narayana, N. K. L., Rahul, S., Supriya, M., 2022. Autonomous Car Controller using Behaviour Planning based on Finite State Machine. In: 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI), 296–302.
- 4) Li, H., Liu, W., Yang, C., Wang, W., Qie, T., Xiang, C., 2022. An optimization-based path planning approach for autonomous vehicles using the DynEFWA-artificial potential field. *IEEE Trans Intell Veh*, 7, 263–272.
- 5) Liang, Y., Li, Y., Yu, Y., Zhang, Z., Zheng, L., Ren, Y., 2021. Path-following control of autonomous vehicles considering coupling effects and multi-source system uncertainties. *Automot Innov*, 4, 284–300.
- 6) Liu, Y., Lyu, C., Zhang, Y., Liu, Z., Yu, W., Qu, X., 2021. DeepTSP: Deep traffic state prediction model based on large-scale empirical data. *Commun Transport Res*, 1, 100012.
- 7) Liu, Y., Wu, F., Liu, Z., Wang, K., Wang, F., Qu, X., 2023a. Can language models be used for real-world urban-delivery route optimization? *Innovation*, 4, 100520.
- 8) Liu, Z., Li, Y., Wu, Y., 2023b. Multiple UAV formations delivery task planning based on a distributed adaptive algorithm. *J Frankl Inst*, 360, 3047–3076.
- 9) Ma, H., An, B., Li, L., Zhou, Z., Qu, X., Ran, B., 2023a. Anisotropy safety potential field model under intelligent and connected vehicle environment and its application in car-following modeling. *J Int Con Veh*, 6, 79–90.

APPENDIX

```
import numpy as np
import os
import sys
import tensorflow as tf
from distutils.version import StrictVersion
from collections import defaultdict
from PIL import Image
import requests
from io import BytesIO
import cv2

from time import sleep
import serial
ser = serial.Serial('COM6', 9600, timeout=1)

ESP32_CAM_URL = "http://192.168.153.31/cam-hi.jpg"

# === TensorFlow Version Check ===
if StrictVersion(tf.__version__) < StrictVersion('1.9.0'):
    raise ImportError('Please upgrade your TensorFlow installation to v1.9.* or later!')

# === Importing Object Detection Utils ===
from utils import label_map_util
```

```

from utils import visualization_utils as vis_util

# === Paths to Model and Labels ===

MODEL_NAME = 'traffic/inference_graph'
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = 'traffic/training/labelmap.pbtxt'

# === Load the TensorFlow model into memory ===

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

category_index =
label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS,
use_display_name=True)

a2 = 0 # for email trigger

# === Detection Function ===

def run_inference_for_single_image(image, graph):
    image_tensor = tf.get_default_graph().get_tensor_by_name('image_tensor:0')

```

```

output_dict = sess.run(tensor_dict, feed_dict={image_tensor: np.expand_dims(image, 0)})

output_dict['num_detections'] = int(output_dict['num_detections'][0])
output_dict['detection_classes'] = output_dict['detection_classes'][0].astype(np.uint8)
output_dict['detection_boxes'] = output_dict['detection_boxes'][0]
output_dict['detection_scores'] = output_dict['detection_scores'][0]
if 'detection_masks' in output_dict:
    output_dict['detection_masks'] = output_dict['detection_masks'][0]

global a2
if output_dict['detection_classes'][0] == 1 and output_dict['detection_scores'][0] > 0.70:
    print('AMBULANCE')
    ser.write("2".encode())
elif output_dict['detection_classes'][0] == 2 and output_dict['detection_scores'][0] > 0.70:
    print('HIGH_TRAFFIC - green')
    ser.write("1".encode())
elif output_dict['detection_classes'][0] == 3 and output_dict['detection_scores'][0] > 0.70:
    print('LOW_TRAFFIC - red')

if a2 == 1:
    a2 = 0
    sleep(1)
    send_email()
    sleep(1)

```

```

return output_dict

# === ESP32-CAM Real-Time Image Processing ===

try:
    with detection_graph.as_default():
        with tf.Session() as sess:
            ops = tf.get_default_graph().get_operations()
            all_tensor_names = {output.name for op in ops for output in op.outputs}
            tensor_dict = {}

            for key in ['num_detections', 'detection_boxes', 'detection_scores', 'detection_classes',
                        'detection_masks']:
                tensor_name = key + ':0'

                if tensor_name in all_tensor_names:
                    tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(tensor_name)

            while True:
                try:
                    # Fetch image from ESP32-CAM
                    response = requests.get(ESP32_CAM_URL)

                    image_np = np.array(Image.open(BytesIO(response.content))).convert('RGB')

                    image_np = cv2.cvtColor(image_np, cv2.COLOR_RGB2BGR) # Convert for
OpenCV display

                    cv2.imwrite('capture.jpg', image_np) # Save for email

                # Inference

```



```

output_dict = run_inference_for_single_image(image_np, detection_graph)

# Visualization
vis_util.visualize_boxes_and_labels_on_image_array(
    image_np,
    output_dict['detection_boxes'],
    output_dict['detection_classes'],
    output_dict['detection_scores'],
    category_index,
    instance_masks=output_dict.get('detection_masks'),
    use_normalized_coordinates=True,
    line_thickness=8)

cv2.imshow('ESP32-CAM Detection', cv2.resize(image_np, (800, 600)))
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
except Exception as e:
    print("Error fetching or processing image:", e)
    sleep(1)
except Exception as e:
    print("Fatal error:", e)
finally:
    cv2.destroyAllWindows()

```