Kalyani Government Engineering College
(Affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal)
Kalyani - 741235, Nadia, WB

Project report

On

Analysis the functionalities of self-driving car using line follower, computer vision and artificial intelligence

(A dissertation submitted in partial fulfillment of the requirements of Master of Technology in Computer Science and Engineering of Maulana Abul Kalam Azad University of Technology, West Bengal)

Submitted by

SARBOJIT DAS
University Roll Number: 10211419001
Session 2019-2021
4th Semester

Under the guidance of
Mr DEBABRATA CHOWDHURY
Associate professor
Department of Information technology

Certificate of Approval

This is to certify that the project report on "Analysis of the functionalities of self-driving car using line follower, computer vision and artificial intelligence" is a record of bonafide work, carried out by Sarbojit Das under my/our guidance and supervision(s).

In my/our opinion, the report in its present form is in conformity as specified by Kalyani Government Engineering College and as per regulations of the Maulana Abul Kalam Azad University of Technology. To the best of my/our knowledge, the results presented here are original in nature and worthy of incorporation in project report for the M.Tech. program in Information Technology.

Signature of Supervisor/ Guide

Professor Debabrata Chowdhury
Asst Professor, Information Technology

Signature of Head, Dept. of IT

Declaration by the student

I Sarbojit Das, a student of M.Tech, IT 2nd year have submitted this report for interim monitoring assessment.

I declare that I have not committed plagiarism in any form or violated copyright while prepared this report.

Name of the student: SARBOJIT DAS

University Roll Number: 10211419001

Sarbojit Das

Full signature:

Date: 16.7.2021

Acknowledgement

First of all I pay my entire gratitude to my guide ***Mr. Debabrata Chowdhury*** *(Asst. prof Dept. of Information Technology, Kalyani Government Engineering College) without his guidance and resources the project would have been impossible to complete.*

Secondly we would like to thank all other faculty members of Department of Information technology who tirelessly encouraged me to do the project and helped me in their every possible way to reach to completion.

Once again thank you to all our teachers and friends to help ne to succeed in this endeavor and grateful for giving us the opportunity to do this project.

Sarbojit Das

Signature

Name of the student: SARBOJIT DAS

Roll No: 10211419001

Abstract

Problem: A driverless autonomous car has shown great success in every domain whether it is the safety of human life or driving the car without having any obstacles. However, analyzing and implementation the functionalities remain an open challenge. Features such as road lane detection, surrounding object detection, environmental sound detection, and classification have some loopholes- inappropriate for real-time analysis, time-consuming, high computation cost, etc.

Methodology: The existing methodologies such as canny edge detector, to detect the road lane are inappropriate for curve lane detection and real-time response. Other existing approaches such as RCNN, sliding windows technique have high computation costs, time consumption, etc. The hypothesis of this work is to incorporate computer vision, artificial intelligence, and a robotic approach to address these loopholes. The hypothesis of this work also initiates to include new functionality to the proposed system

Outcomes: Keeping the above hypothesis in mind, the thesis achieved the following objectives: -1) It detects the road lane using a line follower. A line follower technique finds the lane in the proposed system. Arduino is used to controlling and processing the whole system and, a motor driver drives the motor and, a neural network learns the machine and predicts the output.2) It classifies and localizes all the objects in front of a car using YOLO and helps in decision making.3) It classifies road obstacles using a convolution neural network. 4) It classifies the different environmental sounds detected by the sensors using K Nearest Neighbor. Finally, a performance comparison is done in terms of accuracy between the existing methodologies and the proposed method.

Table of Contents

List of Figure

List of Table

Chapter 1

Introduction

1.1 Overview

Artificial Intelligence plays a crucial role in every sector in the 21st century. It incorporates computer vision and machine learning to provide various applications in different domains. A self-driving car is one of the innovative inclusions of artificial intelligence. An autonomous car is such a type of vehicle that can sense its surroundings and move safely without human intervention. A self-driving car consists of various sensors for perceiving its surrounding –e.g- LIDAR, RADAR, SONAR, GPS, and odometry. Advanced Control systems are used to interpret sensory information to detect navigation paths, obstacles, and other significant indications.
The experiment for an automated driving system (ADS) was started in 1950 but the testing for the fully developed car was not initiated until 1970 by Tsukuba Mechanical Engineering in the laboratory. In 1980, Defense Advanced Research Project Agency(DARPA) provided funds to various universities for the implementation of a self-driving car and their experimentation. Every year almost 1.3 million pedestrians were killed due to road accidents and This statistics is significant because if no measures will not be taken then it will be one of the major causes of human death by 2030. The reason for the development of such technology is to prevent road accidents which are resulted in human death. The fundamental cause of road accidents is human error and carelessness – for example, fatigue, driver drowsiness, misjudgment, high speed, and distraction. The most significant aspect of a self-driving car is to prevent these impending health issues. Human error does not affect the self-driving car. Artificial Intelligence, computer vision, and sensor technologies are used in a driver-less automated car. Recent studies show that a self-driving car (Ruturaj Kulkarni, 2018) would have to be driven millions of miles without having any distraction.

A comparative study between traditional machine learning and a deep learning method has been made. Computer vision is a subdomain of artificial intelligence where a computer can perceive high-level understanding from digital images or videos. This field has gained much popularity due to people's demands and expectations towards this field. This field consists of object detection (Simhambhatla, 2019), object recognition, action recognition, pattern recognition, and automatic guidance. Many research journals have been published about this field especially in machine learning, deep learning, and Convolution Neural Networking(CNN). The CNN has a wide application in different domains such as image processing, video analysis, machine learning, and natural language processing. The main purpose of CNN in computer vision Is object detection and automation based on images or videos. Different CNN techniques are used for object detection such as GoogleNet, Faster R-CNN, Single-shot Multibox Detector(SSD) and You Only Look Once (YOLO).

1.1.1 Self-driving car

A self-driving or autonomous car is a type of vehicle that consists of a camera, radar, and artificial intelligence for traveling from one destination to another destination without human intervention. The purpose of different sensors in self-driving cars is to interpret road lanes, pedestrians, other vehicles, traffic signal.

1.1.2. Main features of self-driving car

The main characteristics of a self-driving car can be summarized in the following ways

1. **Adaptive Cruise Control(ACC)**

Adaptive Cruise Control is also called Dynamic cruise control. It is an intelligent kind of cruise control that automatically controls the speed of the car to adjust the speed with the vehicle in front of it. Adaptive Cruise Control prevents collision. This mechanism completely prevents rear-end collision using severe braking and lane change situations.

2. **Autonomous Emergency Breaking(AEB)**

Autonomous Emergency Braking is one of the improved features for a self-driving car. In this mechanism, the road in front of the vehicle is scanned and applied breaks automatically to prevent a collision.

3. **Blindspot detection**

Blindspot detection is one of the fundamental technologies which offers 360 degrees of digital coverage around a car, irrespective of speed. This technology detects traffic behind the car as well alongside. Blindspot detection can be classified into two types- such as active blindspot detection and passive blindspot detection.

4. **Electronic Stability Control(ESC):**

Electronic Control Stability is one of the core features that controls an automatic computer-controlled braking system to maintain a critical driving situation.

5. **Lane-Keeping Assist**

Lane_Keeping Assists (LKA) is a mechanism in a self-driving car that helps the car to run along a desired lane of the line by controlling the steer of angle. This methodology is used for preventing accidents during free driving.

6. **Reverse Park Assist**

The Reverse Park Assist is used to sense the object behind the blind spot of the car. This tool is used to avoid reverse parking accidents. A simple reverse park assist system consists of an audio warning system, a camera, and a video monitoring system. This tool uses various sensors located rear bumper of the car.

7. **Rear Cross-Traffic Assist**

Rear Cross-Traffic Assist is used to reverse out the perpendicular parking region when their rear view is blocked. This system has two mild range sensors in the rear of the car for measuring and interpreting the distance, speed, and locus of the cars tracked in cross-traffic.

8. **Traffic Jam Assist**

Traffic Jam assist is an extension of the Adaptive Cruise Control. This mechanism is a low-speed version of ACC which controls the speed considering other vehicles. Traffic Jam Assist depends on the sensors and working of the Adaptive Cruise Control with going and stop. If the ACC with stop and go is resumed, it continuously interprets the velocity of the surrounding vehicles and compares it with its velocity.

9. **Vehicle to Vehicle Communication**

Vehicle to Vehicle communication is a method that shares information regarding the velocity, distance, and position of the surrounding vehicles. The underlying technology of V2V communication helps vehicles to broadcast and receive Omni-directional information.

10. Vehicle Guidance System

This technology allows the vehicle to drive on the road without human intervention. The vehicle guidance system contains a sensor fusion model, a path generator, and a motion planning algorithm.

1.1.3. How self-driving car works?

Artificial Intelligence enhances the performance of self-driving cars. The working of a self-driving car depends on sensors, complex machine learning algorithms, actuator, powerful processor, and GPU to run the system. The system developers use a large amount of image data, along with deep learning and neural network to implement a system that can travel automatically.
The neural network is used to identify the patterns in the data, machine learning algorithms are applied to this data. These data are images that are taken from an autonomous camera on an autonomous car from which the neural network trains to detect and classify various objects such as pedestrians, cars, trees, traffic signals, animals, etc.

Various sensors are located in different parts of the vehicle. Radar sensors are used to monitor the location of the car. The video camera is used to capture the road lane, traffic signal, traffic light, other vehicles, pedestrians. Lidar(Light Detection and ranging) uses light pulse to calculate distance, find the road edges and track the lane markings. Ultrasonic sensors in the wheel are used to find the curbs and other vehicles Software then processes all the input, defines a path, and transmits an instruction to the car's actuator which is used to control the brakes, steering, and acceleration.

1.1 Research question

The following research questions can be raised for this given study

1. What self-driving car is so important?
2. How do self-driving cars ease human limitations?
3. How do the previous studies' gaps overcome?
4. Why line-follower and computer vision are necessary for the proposed system?

1.3 Motivation

Human life is a precious gift of god and it needs to be saved. Every year a huge number of pedestrians and passengers died due to driver's negligence. This record is continuously rising and it is an alarm to take the right decision. This issue pushes to allow autonomous cars to operate on the road. However, a recent Uber Institution self-driving car accident that killed a pedestrian raises the question of whether this type of vehicle is ready to travel on the public road or not. In this study, we analyze this issue and propose a solution for this problem. State-of-the-art computer vision and robotics play a great role to resolve this issue. Robotics enables car and passenger safety during travel and image processing, object recognition, and object detection in low light ensure pedestrian safety.

The main motivation for implementing such a system is to ease the problems and loopholes remaining. The line follower system does not depend on the color segmentation and edge detection and it does not require to operate on the straight line, it can also work on the curvature. YOLO algorithm provides the highest accuracy for classification and localization of the object using a pre-trained model. A convolution neural network provides an effective result for classifying the obstacles ahead of a vehicle. Finally, KNN supervised learning achieves a great amount of accuracy for the classification of urban data.

1.4 Objective

In this research on the core functionalities of self-driving, we can find the four main functionalities of an autonomous car. The main objectives of this research can be summarized as follows

- Finding the track of the car on which it travels using line follower and IR sensor technology.
- Detect and classify the objects such as pedestrians, cars, animals, traffic lights, road signs ahead of a vehicle using the You Only Look Once (YOLO) algorithm.
- Classification of road obstacles such as barricade or a guard rail, fire, fog, and potholes using Convolution neural network and predict the accuracy and occurrence of the object.
- Classification of environmental sounds using K Nearest Neighbor and predict the accuracy and occurrence of the object.

1.5 Organization of the thesis

The thesis consists of five interrelated chapters presented below.

Chapter 1is the introductory chapter that presents overview of the research problem, motivation of the research, summarizes the methodologies used in coming chapters and explores the required resources.

Chapter 2 serves as a literature review of the previous methods that identify research problem, explores the relationship between different methods selected for the work and the previous methods and provides some preliminary outcomes to justify the scope and approach of this work.

Chapter 3 highlights different methodologies that are applied to analysis the various functionalities of the proposed system and explores the scope of the system.

Chapter 4 presents the result of different methodologies that are applied in the previous chapter. It also compares the performance of the proposed system with the existing system.

Chapter 5 concludes the thesis, summarizes the work and outcome and highlights scope of future work

1.6 Hardware and Software used

Hardware: IR multipurpose sensor, ultrasonic proximity sensor, LIDAR, video camera, Arduino uni, Motor driver

Software: Jupyter notebook with python 3.6.

Chapter 2

Literature Review

Over the century, various researchers conducted on the features of a self-driving car. Multiple pieces of research have examined trends, areas of research, and methodology in a general and specific domain of science and technology. According to wall journals, the study for the self-driving car was initiated in the 1920s when engineering professionals implemented a plan for Automated Driving System(ADS) and the experiment was started in the 1950s. However, Tsukuba Mechanical Engineering Lab in Japan first developed tested automated vehicles in the 1970s. This type of system consists of two cameras and an analog computer that will work together to generate the images of the roadway. In 1991, the USA invested $650 million for the automated national highway system which used special technology for guiding the self-driving vehicle along a stretch of a road. In 2015, Delphi technology helped engineers to use an improved methodology for self-drive vehicles - "self-drive cars" and the Audi model. The research for a self-driving car has been conducted in three folds- road lane detection, object recognition and classification, and sound detection.

2.1.Lane detection

The first fundamental feature for a self-driving car is road lane (Abdulhakam.AM.Assidiq, 2008) detection. Various researchers have applied various methodologies for detecting the road before running the car on the road. International Islamic University In Malaysia proposed a well-known road lane detection approach- canny edge detection. John F Canny proposed this popular approach in the 1980s. The approach of canny edge detection is mainly
highlighted in PM Daigavane, PR Bajaj conference paper for road lane detection. The classical canny edge detection approach is the most popular method than any other edge detection method. In this algorithm, images are captured and converted to grayscale. Then, the F.H.D algorithm is applied to remove noise for efficient edge detection. An edge detector is used to determine the location of the lane boundaries. A sharp contrast between the surface of the road and the colored line defines the lane boundaries. The line detector is applied with a restricted search space. The canny edge detector is used to extract the structural information from the road images and optimizes the amount of processing data. This calculus of variation is used to find a function that optimizes a given function. The sum of four exponential functions is used to define the optimal function in a Canny edge detector. Another lane detection approach is an instance segmentation approach which consists of a lane segmentation branch and a lane embedding branch. In this approach, LaneNet is used as a network that links binary segmentation with the clustering loss function. In the output section, each lane pixel is allocated to its corresponding lane. National Chiao Tung University proposed a road lane marking detection using deep learning and convolution neural networks. This methodology is divided into three parts- Lane marking generation, Lane grouping, and Lane Model Fitting. CNN semantic segmentation (Thi Nhat Anh Nguyen) (Brilian Tafjira Nugraha, 2017) is used to generate lane marks. Land Mark Detection (LMD) system is trained with a three-class dataset- road, lane, and others. After detection of the lane, images are fetched. Lane grouping follows two steps- firstly, neighbor pixels belong to the same lane segment are clustered to generate super marking. The connected component labeling method is applied to find connected regions and allocate each value to each region. In the next step, supermarkings are connected on the same lane marking. The measurement function is designed properly so that it can calculate the cost of connecting super marking. Both straight lines and curve lines can be represented by Lane Fitting Model. Another approach for lane detection for a self-driving car is a hybrid deep-learning Gaussian process (DL- GP) architecture. In this method, pedestrian lane detection is cast in an unstructured environment as a segmented problem where the tested image is divided into the pedestrian lane and background regions. This architecture consists of a compact convolution encoder-decoder network and a hierarchal GP classifier. This approach is preferable because it consists of a little number of parameters having a non-parametric GP classifier.

2.2.Object detection for self-driving car

The second crucial feature for a self-driving car is to detect and classify the object in front of them. The most classical approach for object detection is the sliding window technique and this method has been working for a long time. This algorithm is best suited for a linear and straightforward classifier. The sliding window technique is based on two parameters- extracting a feature from an image window and making a decision based on the classifier. Another approach for object detection is a single shot multibox detector (Hyunggi Cho, 2014) in which object localization and classification can be dealt with a single forward network. In this approach, a single forward network is used to generate a fixed-size set of bounding boxes and scores for the occurrence of object class instances in those bounding boxes.

The next step is non-max suppression to generate final detection. One of the popular object detection approaches is a region with convolution neural networks(R-CNN) which is used to combine rectangle or square proposals with CNN features. RCNN follows three steps- i) identify the regions (Walzel, 2018) in the image of the object. These are known as region proposals. ii) Fetch CNN features from the regions. iii) classification of the objects by extracted features. There are three types of RCNN. Girshik propose (Chan Yee Low, 2014) RCNN detector that produces region proposals by the Edge Boxes algorithm. Images are cropped out and resized in the proposed region. Then CNN is used to classify the cropped and resized cropped images. Finally, SVM which is trained using CNN is applied to the region proposal bounding box. The working of fast RCNN is similar to RCNN, the only difference is that fast RCNN processes the whole image rather than cropping and resizing images. Fast RCNN is used to pool CNN features corresponding to each region proposal. This algorithm is much faster than RCNN and it uses shared overlapping regions. Faster RCNN includes a Region Proposal Network(RPN) to implement region proposals directly to the network. Another kind of CNN is a Region-Based Fully convolution network (RFCN) which generates accurate object detection and efficient result. RFCN is almost analogous to faster RCNN, the only difference is that faster RCNN crops from the same layer where the region is predicted whereas RFCN crops the last layer of the feature before the prediction.

2.3. Sound classification

Sound detection and classification is another important feature for a self-driving car. Supervised and unsupervised both machine learning approaches are used for sound classification. Center for Urban Science and Progress, New York and Music and Audio research laboratory, New York proposed spherical K Mean algorithm for classification of urban sound. The main difference with classical K means clustering (Justin Salamon) is that the centroid for each cluster lies on the unit sphere. This algorithm uses different techniques such as spars (Ping-Rong Chen, 2018)e coding which helps to learn features from audio. The most common approach for urban sound classification is the convolution neural network approach which contains two convolution layers with max-pooling and two fully connected layers. The accuracy of the network is measured by three publicly available datasets. School of Mechanical Engineering, Jiangnan University in china proposed a classification approach based on 2 order dense convolution neural network (Iurii Lezhenin, 2019) using a dual network. This algorithm is applied under a noise environment and generated effective results. Another popular approach for sound classification is classification using convolution recurrent neural networks (CRNN). In this method, the network takes waveform as an input in the field of sound classification. Convolution recurrent neural network contains convolution neural network for fetching sound features and recurrent neural network for gathering extracted features. Long Short Term Memory Neural Network (LSTNM) (Jonghee Sang, 2018) was proposed by Peter the Great St.Petersburg Polytechnic University St.Petersburg, Russia, and University of Aizu Aizu-Wakamatsu, Japan. LSTM is trained on mel-spectrograms fetched from an audio dataset. Fivefold cross-validation is used to measure the performance of the model and compare it with baseline CNN.

Chapter 3

Methodology

The current research aims to examine the different functionalities of a self driving car using artificial intelligence and robotic technique. Computer vision, machine learning and deep learning incorporate with line follower to overcome previous research gaps and provide an appropriate solution. The research also focuses to explore the different functionalities of a self driving car such as automatic road detection, automatic car, pedestrian and signal detection, classification of road obstacles and detection and classification of environmental sound

The methodologies of this proposed system can be summarized as-

1. Road lane detection using line follower
2. Surrounding object detection using YOLO algorithm
3. Classification of road obstacles using CNN
4. Environmental sound detection and classification using KNN

3.1.Road lane detection using Line follower

Line follower is a popular technique that is used in robots. The working principle of line follower is depended on automation that follows a particular direction, generally black line on a white surface. The line follower has two sensors that are installed at the frontend and DC motors drive the wheels. The control section present in the line follower controls the speed of the wheel according to the signal received from the input.

3.1.1Definition of Line follower

A line follower robot is an autonomous robot that tracks the line and follows that line, proceeding along a fixed track that can be varied by moving the lines. A line follower robot senses the color, generally, it senses the white line on the black surface or the black line on the white surface. The line follower consists of two sensors to sense the white color. It contains a DC motor to control the movement of the robot.

3.1.2Features of a line follower

•A line follower is always following a specific direction or path and according to the situation, it maintains its working nature.

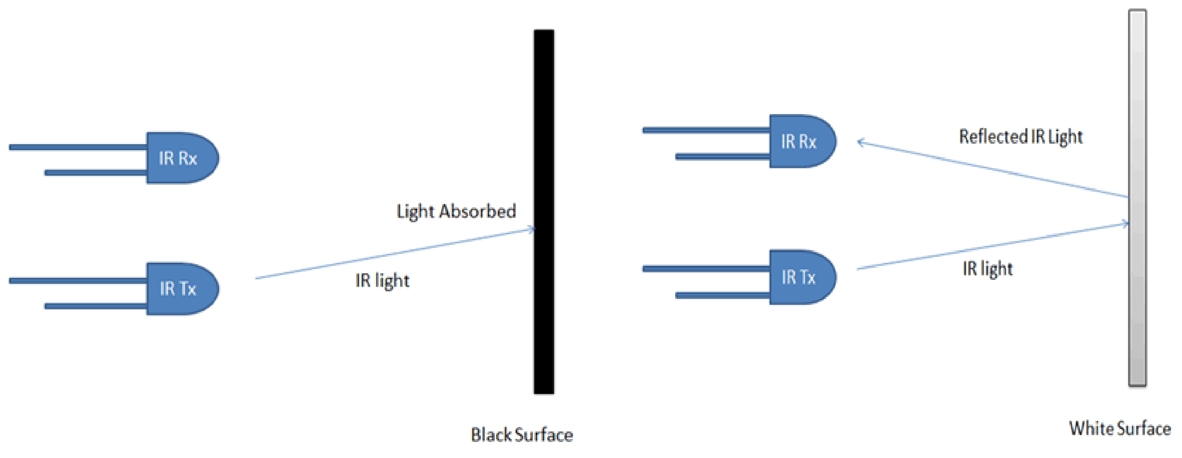
•A line follower always maintains a feedback logic where the track is generally black line on the white surface.

•A line follower always maintains the same path and it never gets deviate from the track

•Line followers are two types- mobile-based line follower and RF-based line follower

3.1.3. Basic principle of Line follower

The concept behind the line follower is associated with light. The behavior of light on the white and black surface is used to build a line follower. When light incidents on a white surface it completely reflects but for a black surface, light is absorbed completely.



(i) (ii)

Fig: 1.1: Incidence of the light on (i) black surface Fig (ii) on white surface

In this line follower, IR transmitters and IR receivers are used and these IR transmitters and IR receivers are also known photodiodes. IR transmitters and IR receivers send and receive light. IR sensor is used to transmit infrared light. When infrared ray incidents on the white surface, it will reflect and be collected by a photodiode which produces some voltage variation. When an infrared ray incident on the black surface, the black surface absorbs the light, as a result, the diode will not receive any light or ray. Arduino line follower regards 1 as an input when a sensor senses white surface and 0 as an input when a sensor senses black surface.

3.1.4 Components of Line follower

The components of the entire line follower are shown below

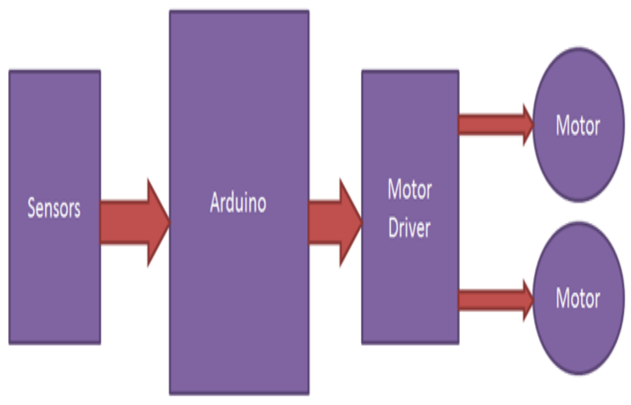


Fig 1.2: Block Diagram of Line Follower

The line follower consists of three submodules:- a sensor section, a control section, and a driver section.

- **Sensor section:** The main components of the sensor section are IR diodes, comparator(op-Amp), potentiometer or voltage calculator, and LEDs. The main purpose of the potentiometer is to set a reference voltage at one comparator's one terminal and IR senses the line and provides a variation of the voltage at another terminal. Then, the comparator is used to compare both voltage and produce a digital signal at the output.

- **Control section:** Arduino controls the entire process of the line follower car. The output of the comparator is fed into the digital pin of the Arduino. These signals are read by the Arduino and Arduino delivers a command to the driver circuit.

- **Driver section:** The main components of the driver section are the motor driver and two motors. The motor driver is used to drive motors, and the motor driver receives commands from Arduino to drive the motors.

3.15. How neural network works in the backend

An 8 bit IR sensor is used for detecting the line in the proposed system. The diagrammatic representation of the IR sensor with neural network is shown below

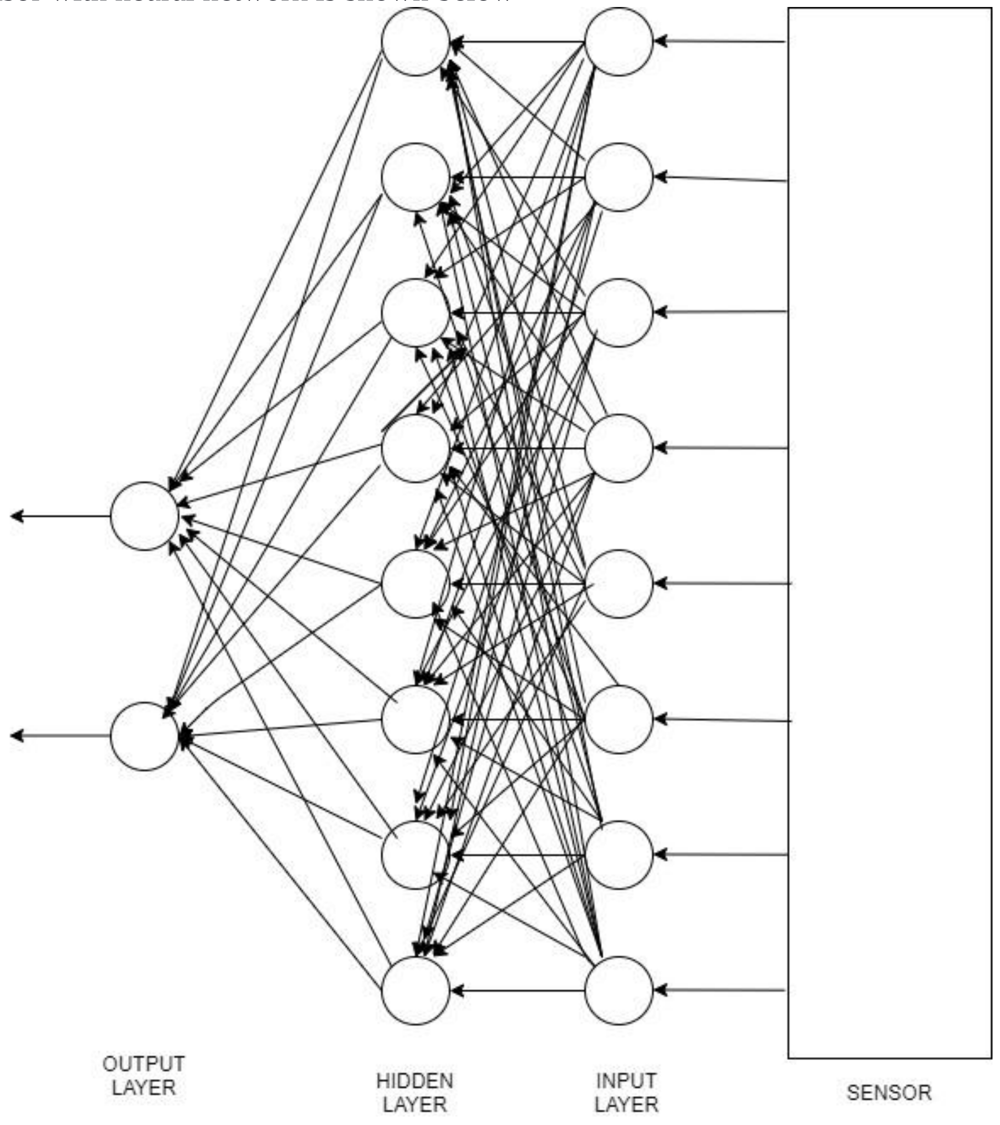


Fig 1.3: IR sensor with neural network

The 8bit sensor is used to provide the input to the model. In this proposed system, one input layer, one hidden layer, and one output layer are present.

The working of the above network are given below

- The first step is to check the input size and output size.

- After initialing, all the inputs are connected with all the nodes in the hidden layer. Every connection will weight it. Therefore, weights have to initialize for every connection.

- After the initialization of weight, the data starts to flow. For the particular input, there has been a particular output that has been provided by default.

- After input has been given to all the nodes, each input is multiplied with weights, and bias is added with the result and gives the output.

- These outputs will be activated using the activation function when they come to the hidden layer.
  The equation is given below
  Activation function= f(z)
  And $z=\sum_{1}^{n} wi\ xi + bi$
  Where xi = input at a node i
  wi= weight associated with node i
  bi=bias associated with node i

•Outputs of the hidden layer go to the output layer to calculate the cost. The cost will decide whether the expected output differs from the actual output. The data flow from input to output through the hidden layer to compute the cost is known as forward propagation.

•According to the cost, weights are updated going backward is known as backward propagation.

•Forward and backward propagation continuously work together and after while neural network starts to learn such as for the particular input, there will be particular output

•After that, the neural network starts to generate output for particular input by involving the same bias weight and bias. This way proposed system starts to predict the output.

3.1.7 Flowchart

The flowchart for the output1 and output 2 is given below

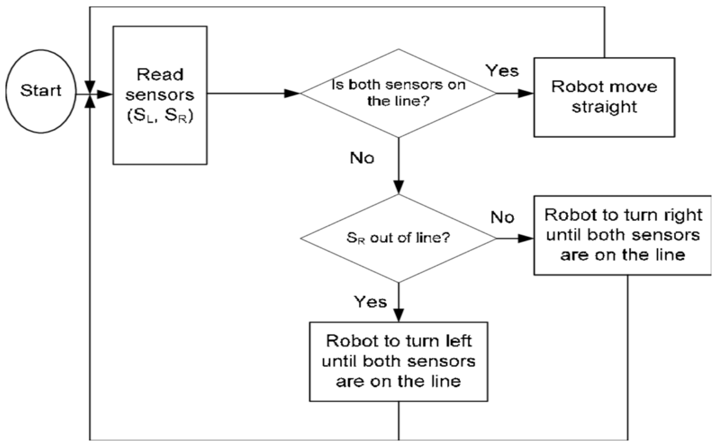

Start
Read sensors (S_L, S_R)
Is both sensors on the line?
Yes
Robot move straight
No
S_R out of line?
No
Robot to turn right until both sensors are on the line
Yes
Robot to turn left until both sensors are on the line

Fig 1.4: Flowchart for the output of the sensor

3.18. Algorithm

Table 1.1: Algorithm for output of the sensors using line follower

```
if $S_1$==1 AND $S_2$==1
        Vehicle moves forwards
else if $S_1$=0 AND $S_2$==1
        Vehicle turns left
else if $S_1$==1 AND $S_2$==0
         Vehicle turns right
else
        Vehicle stops
```

3.2 Object Detection Using You Only Look Once(YOLO)

YOLO is one of the popular algorithms for real-time object detection. YOLO uses a single convolution network that is used to predict bounding boxes and class probabilities. The bounding boxes are weighted by the probabilities and the model predicts based on the final weight.

3.2.1 Principle of YOLO

- You Only Look Once(YOLO) is used to frame object detection in images as a regression problem to spatially divided bounding boxes and corresponding class probabilities.

- In this algorithm, a single convolution neural network separates the images into the region and predicts bounding boxes and probabilities for each region.

- The neural network is used to predict bounding boxes and class probabilities from entire images in a single run.

- The base YOLO model is used to process images at a rate of 45 frames per second.

3.2.2 Properties of YOLO

YOLO algorithm is based on the following parameters

Bounding Box

A Bounding box is a virtual rectangle box that indicates spatial location of an object. A bounding box is implemented around any object to serve as a point of reference for that object and generates a collision box of that object. The bounding box prediction consists of 5 components-
(x, y, w, h, confidence).
The (x, y) coordinates denote the center of the box for the location
of the grid.
The (w, h) denote the width and height of the bounding box.
Confidence denotes existence of the specified object in that box
The normalized value for (x, y) lies between 0 to 1. The normalized value for (w, h) with rest to image size lies between 0 to 1.

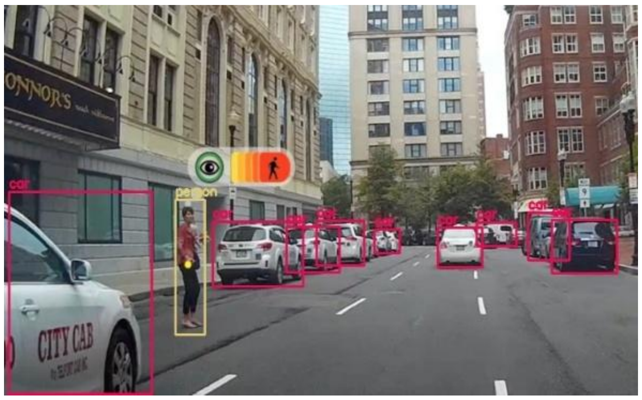
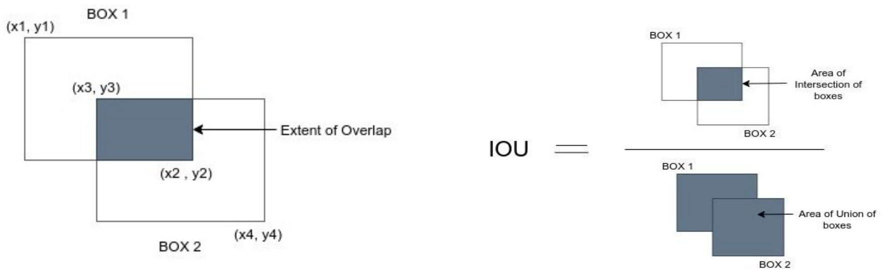


Fig 1.5: Bounding Box for YOLO

Intersection over Union:

It is an evaluation metric that evaluates the accuracy of an object detector. It describes the overlapping of two boxes. The main purpose of IOU is to detect an object for which a model is trained to generate a box that fits around the detected object. It is also applicable in non-max suppression, which removes multiple boxes surround the same object.

If a box1 denoted by [x1, y1, x2, y2] and box2 is denoted by [x3, y3, x4, y4]



(i) (ii)

Fig 1.6: (i) overlapping of two boxes (ii)Intersection over Union for YOLO

$$\text{So, IOU}=\frac{\text{Area of Intersection of two boxes}}{\text{Area of Union of two boxes}}$$

- **Non-max suppression:**

Non-max suppression is a computer vision algorithm used in object detection where the best bounding boxes are selected out of a set of overlapping box. The criterion for non-max suppression is to discard the entities that are below a threshold probability(in general 0.5) bound. The entity that has the highest probability is selected from the remaining entity.

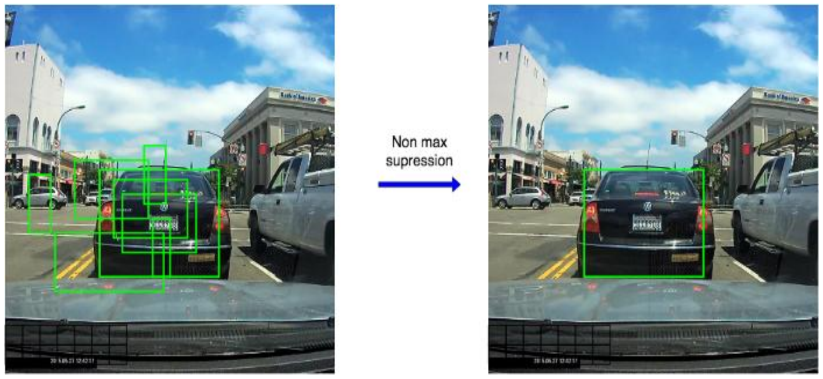


Fig 1.7: non max-suppression for YOLO

- **Prediction vector:**

  Prediction vector is an array which consists of multiple parameters such as probability of the bounding box, coordinate of the bounding box, dimension of the bounding box and different classes. The prediction vector can be defined in the following way

Y=

P_e
b_x
b_y
b_w
b_h
C_1
C_2

Where
Pe is the probability of bounding box containing an image
bx ,by are the x and y coordinate of the bounding box
bw,bh is the width and height of the bounding box
C1, C2 are the probabilities the cell contains an object that belongs to class 1 or class 2 given the bounding box contains an object.

- **Class probabilities or Confidence score:**

  The confidence score is known to be a threshold that finds out the lowest matching score acceptable for detection. The value of confidence score lies between 0 to 1 and each detection describes the confidence and probability of a detected object that belongs to a specific class. Mathematically confidence score is defined as

  Confidence score= *Pr(Object) * IOU(pred, truth)*
  Pr(object)= probability of an object belong to a particular class.
  IOU= Intersection over the union.

  Example- The input image consists of an (S X S) grid of cells and each grid of cells is responsible for predicting the presence of the objects. Each grid of cells is used to predict the bounding box and class probabilities. The shape of each cell from the grid will be C+B*5, C denotes the number of classes, B denotes the number of bounding boxes. Here % is used as a multiplier because it considers 5 parameters(x,y,w,h,confidence).

  The shape of overall prediction is equal to S X S X (C+5XB)

Loss Function:

Yolo calculates loss function by using sum squared error between the prediction and ground truth.

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

∑^B indicates the gross total of the loss values for all the bounding boxes in the cell with respect to its centroid_x, centroid-y, width, height & confidence score,

∑^S² indicates the gross total of all the loss values amongst all the cells in the output feature map.

1^obj represents the occurrence of the object. If the object is present in the cell then it will be 1 otherwise 0.

1^noobj represents the absence of the object. If no object is present in the cell, then it will be 1 otherwise 0.

λs are constants.

The loss function consists of the following loss

- Classification loss
- Localization loss(error between ground truth box and predicted bound box)
- Confidence loss

Classification loss:

The classification loss can be calculated in the following way

$$\sum_{i=0}^{n} 1\text{\^{}obj} = \sum_{c\epsilon \text{classes}} (\text{Pi}(\text{c}) - \text{pi}(\text{c}))\text{\^{}}2$$

If the object presents in the cell, then the squared error of conditional class probabilities for each class
$_{1i}$^obj =1 if an object appears in cell I, otherwise 0

pi(c) denotes the conditional class probability for class c in cell i

Localization loss:

Localization loss can be determined by the difference between an error in bounding box centroids and their width and height.

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$

where

$\mathbb{1}_{ij}^{obj} = 1$ if the j th boundary box in cell i is responsible for detecting the object, otherwise 0.

λ_{coord} increase the weight for the loss in the boundary box coordinates.

Confidence loss:

Confidence loss can be determined in the following way

$$\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

where

$\mathbb{1}_{ij}^{noobj}$ is the complement of $\mathbb{1}_{ij}^{obj}$.

$\hat{C}_i$ is the box confidence score of the box j in cell i.

λ_{noobj} weights down the loss when detecting background.

Confidence loss is calculated twice one for 1^obj and other for 1^noobj.

3.2.3 Dataset:

YOLO algorithm uses the Common Objects in Context (COCO) dataset to train the model. This pertained model uses as a weight to detect the surrounding objects. This dataset has 121,408 images, 883,331 object annotations, 80 classes, super pixel stuff segmentation and 91 stuff categories.

3.2.4. Surrounding objects detection for self-driving car using YOLO

YOLO is a regression-based algorithm rather than choosing the interesting part of the image

The steps of the YOLO algorithm are as follow-

- The input image is taken by YOLO and the size of the input image is 800 X 416.
- Then, the input image is sent to CNN which generates an output having dimensions 19 X19 X5 X85.
- Each grid dimension is 19 X19.
- Image classification and localization are applied on each grid.
- Non-max suppression is applied to eliminate the redundant and overlapping boxes.
- YOLO is used to predict the bounding boxes and corresponding class probabilities for objects.

3.2.5 Flowchart

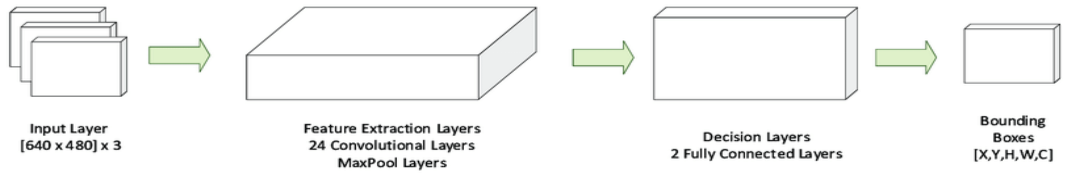


Fig 1.8: Flow chart for object detection using YOLO

3.3 Classification of road obstacles using Convolution Neural Network

Convolution Neural Network (CNN) is a particular type of feed-forward artificial neural network in which the visual cortex is responsible for the connection between neurons. CNN is also called ConvNet is a class of neurons that is used to process data having grid-like topology for example image

3.3.1 Basic architecture of Convolution Neural Network

A Convolution Neural Network consists of three types of layers – convolution layers, pooling layers, and fully connected layers. A CNN architecture is built when all these layers are assembled. Apart from, these three layers, two other essential parameters- activation function and drop out.

The architecture of the Convolution Neural Network is shown below

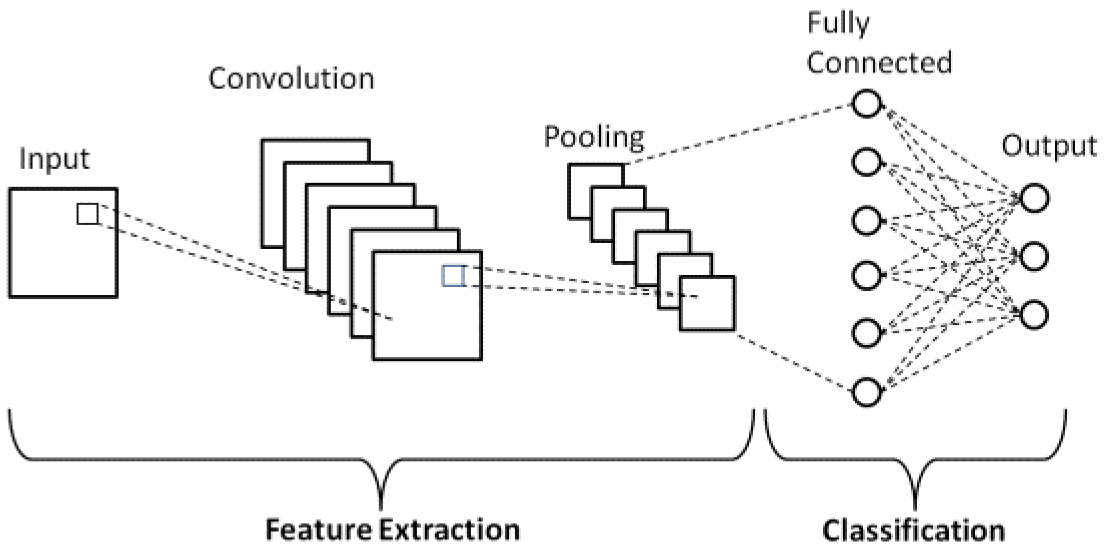


Fig 1.9: Basic architecture of Convolution Neural Network

CNN architecture has two main parts –

•Feature Extraction which is used to separate and identify the different features of the image for analysis.

•Fully connected layer that takes the output from the convolution process and predicts the label of the image depending upon the previous feature extracted in the last stages.

Convolution Layer:

This layer is the fundamental building block of Convolution Neural Network. This is the first layer that extracts different features from the input images. In this layer, the mathematical operation of convolution between the input image and a filter or Kernel of a specific size SxS is done. The dot product is performed between the kernel and section of the image w.r.t size of the image SxS by sliding the kernel over the image.

If the dimension of the image matrix be (h x w x b) and the dimension of the filter be (fh x fw x d), then the dimension of the output image (h-fh +1) x (w-fw+1)x1

For example, consider an image having dimension (5 x 5) whose pixel values are 0 and 1 and a filter matrix having dimension 3x3.

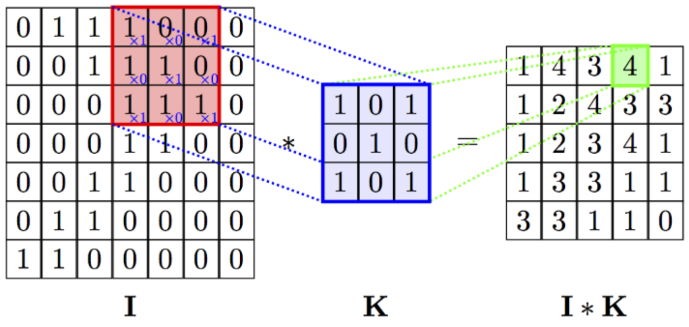


Fig 2.0: convolution of a (5 x5) images with a(3x3) filter

Then, the convolution of a (5 x 5) image with a (3 x 3) filter (kernel)generates a feature map.

Three hyperparameters may affect the volume of the output –

• The depth of the output can be affected by the number of filters. For example, three different filters would generate three future maps, creating the output of three.

• Stride is measured as a distance or number of pixels that can move over the input matrix.

• When filters are no longer fit into the input image, zero-padding is rendered. All the elements that lie beyond the input images are set to be zero and generating a larger size output. There are three types of padding

- **Valid padding**: It is also called no padding. If dimensions are not aligned, then the convolution layer will be dropped.
- **Same padding:** This padding ensures that the size of the output layer is equal to the input layer.
- **Full padding:** This padding is used to increases the size of the output by including zero at the border.

Pooling Layer:

In a convolution neural network, a convolution layer is followed by a pooling layer. This layer is also called downsampling. The primary function of this layer to reduce the size of the convolved feature map to reduce computation cost. This is done by reducing the connections between layers and independent operations performed on each feature map. This reduces the dimension of input, the number of parameters in the input. In the pooling layer, the kernel uses an aggression function to the values within the receptive field and maximizing the output array.

There are two types of pooling layer

Max-Pooling:
In this type of pooling, when the filter moves across the input, it chooses the pixel with maximum value to deliver to the output array.

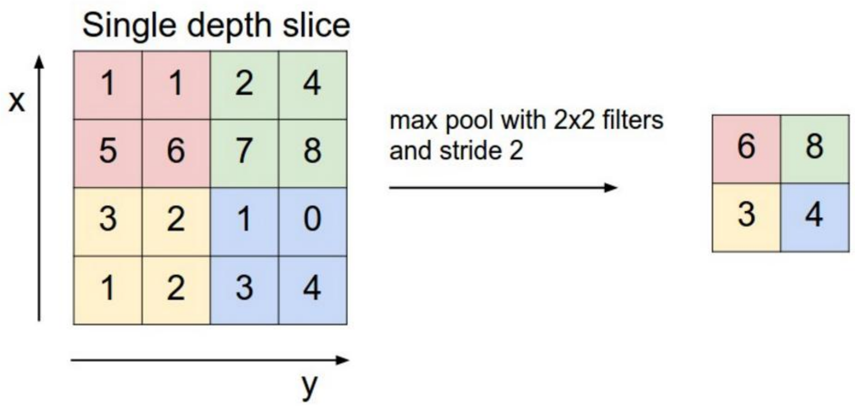


Fig 2.1: Max-pooling using a 2x2 filter and stride=2

Average-Pooling:
In this type of pooling, when the filter moves across the input, it calculates the average value to deliver to the output array

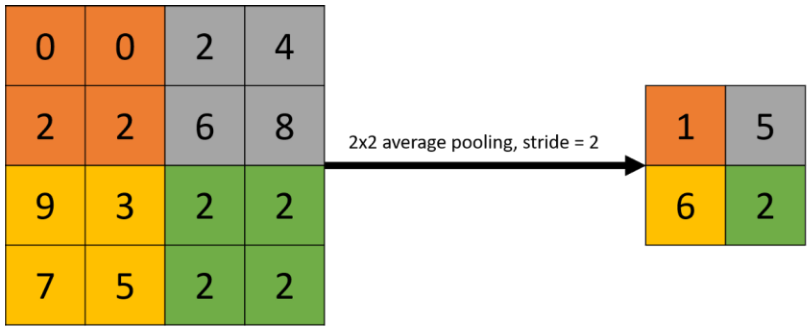


Fig 2.2: Average-pooling using a 2x2 filter and stride=2

Fully Connected Network:

A Fully Connected (FC) network contains the weight and the bias along with neurons and it connects neurons between two distinct networks. These layers are located before the final output layer and build the last few layers of the CNN architecture.
These layers are responsible for performing classification tasks depending upon the features extracted through the previous layers and their various filters. Convolution and pooling layers use relu function but FC layers use the softmax function for classification.

There are two important parameters for neuron networks are Activation function and dropout.

Activation Function:

The activation function in a neuron network is the mathematical function that finds out the output of the network model. It is used to decide whether a neuron needs to be activated or not by measuring the weighted sum and further including bias with it. The activation function normalizes the output of any input in the range of -1 to +1 or 0 to +1.
Mathematically, activation function can be written as

$$Y = \text{Activation function}(\sum (\text{weights}*\text{input} + \text{bias}))$$

Feature of activation function

- Nonlinearity
- Continuously differentiable
- Range
- Monotonic
- Absolute identity

Type of Activation function

- Linear Activation Function
- Non linear Activation Function

Linear Activation Function:

A linear activation function is an equation of a straight line. Therefore, the output of the function will not be restricted in any range. The range of this function is –infinity to + infinity.

The equation of linear activation function is given by

$$Y=f(x)$$

The plot of the linear activation function is shown below

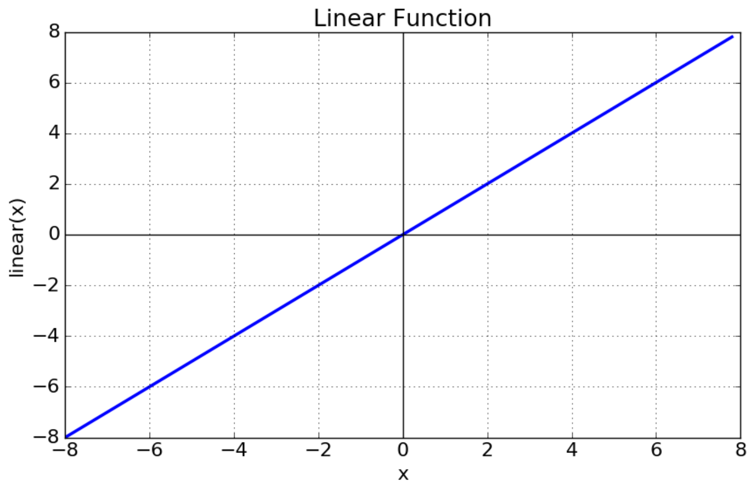


Fig 2.3: Curve for linear activation function

Non-Linear Activation Function:

Non-linear functions are mostly used by modern neural networks. By using nonlinear activation function, models are able to create a mapping between the network's input and output.

Different types of Non-linear activation functions are given below

Sigmoid function:

This function is a differentiable, confined, real function that is defined for all real inputs and does not have a negative derivative at each point. The sigmoid curve is an 'S' shaped graph. The range of a sigmoid function is 0 to 1.

The equation of sigmoid function is given below

$$Y = 1/(1 + e^{-x})$$

The plot for a sigmoid graph is shown below

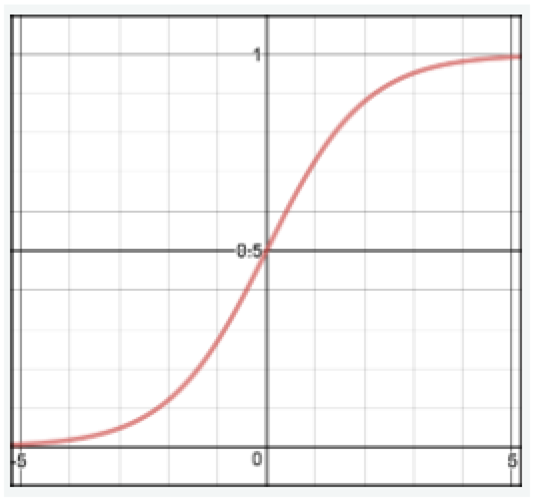


Fig 2.4: Curve for sigmoid activation function

Tanh function:

This function is differentiable, monotonic and continuous in nature. This function is an extension version of sigmoid function. The range of this function is -1 to +1. The equation of this function is given below

$$Y=\tanh(x)$$

$$\tanh(x) = (e^{x} – e^{-x}) / (e^{x} + e^{-x})$$

The plot of this function is shown below

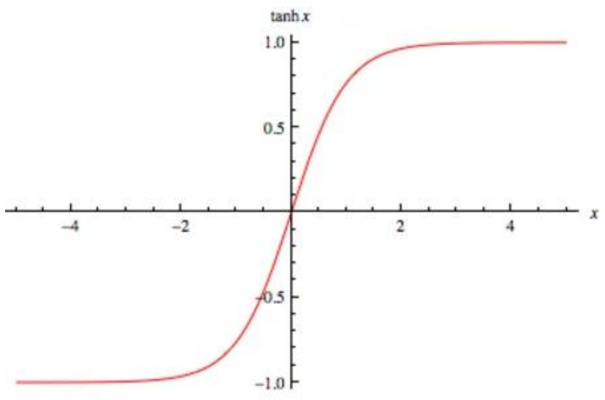


Fig 2.5: Curve for Tanh activation function

RELU function:

A RELU function or Rectified Linear Unit function is mainly used in a neural network. RELU function is implemented in the hidden layer. The range of a RELU function is 0 to infinity.

The equation of a RELU function is given below

$$f(x)=\max(0,x)$$

If the input is x, the value of a RELU function is x and if the input is x, the value of a RELU function is zero.

The plot of a RELU function is shown below

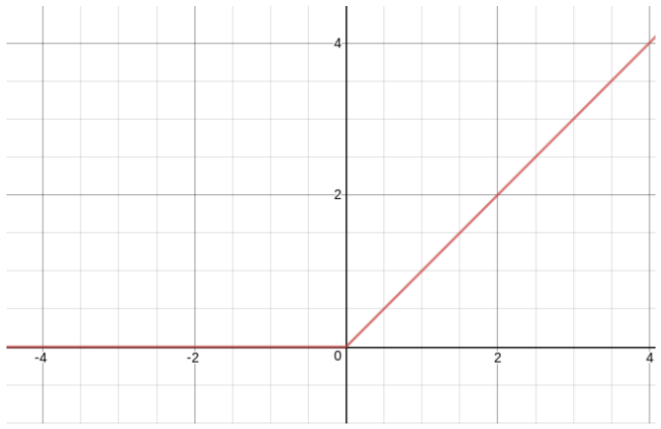


Fig 2.6: Curve for RELU activation function

Softmax:

The softmax function is also known as a soft argument function or multi classic logistic regression. This is a type of logistic regression function for handling classification problems. The softmax function can be applied as a classifier for mutually exclusive classes.

The equation for the soft max function is shown below

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

The plot for the softmax function is shown below

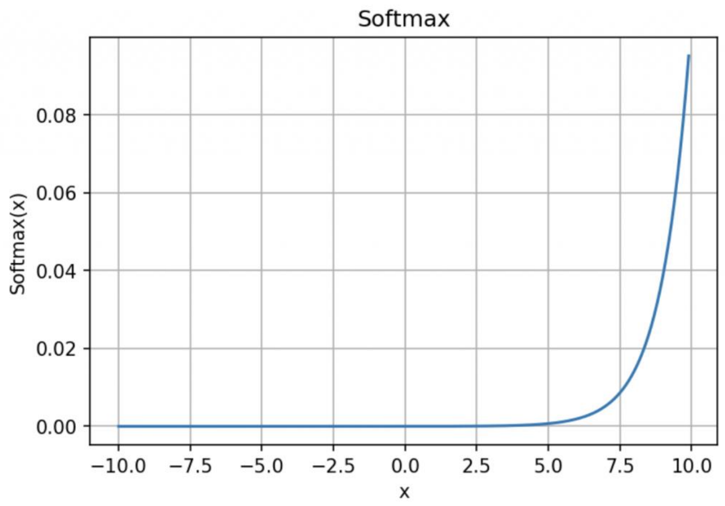


Fig 2.7: Curve for softmax activation function

Drop out layer:

Dropout is a regularization method that prevents overfitting in the model. In the dropout technique, some nodes are dropped out from the network. The dropout layer is used to nullify the activities of some neurons towards the next layer and remain unaltered with others. The dropout layer can be applied on the input end to nullify some features of it. It can also apply to the hidden layer to nullify some features of it. Dropout layers are on the training data to prevent the overfitting problem.

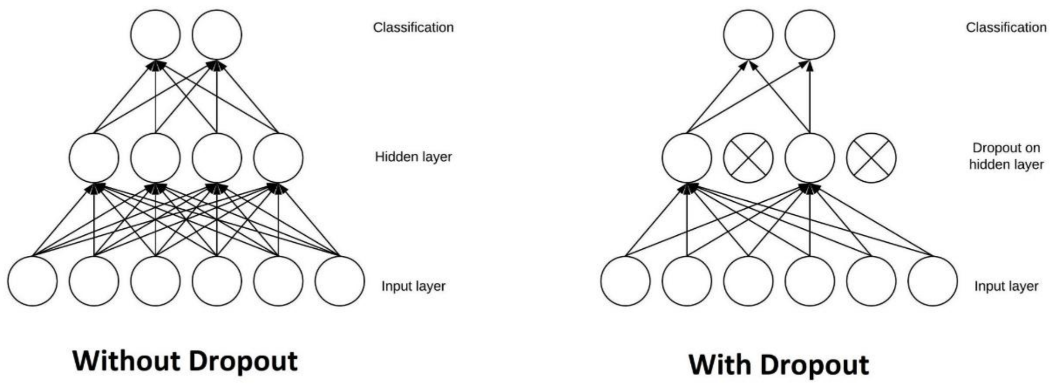


Fig 2.8: Comparison of without dropout and with dropout

3.3.2. Dataset:

To analysis, the road obstacles different images for obstacles were taken. Four obstacles are considered that can block the road. These obstacles are barricades, fog, fire, and potholes. Total 1200 images are taken and each obstacle has 300 images for data preprocessing.

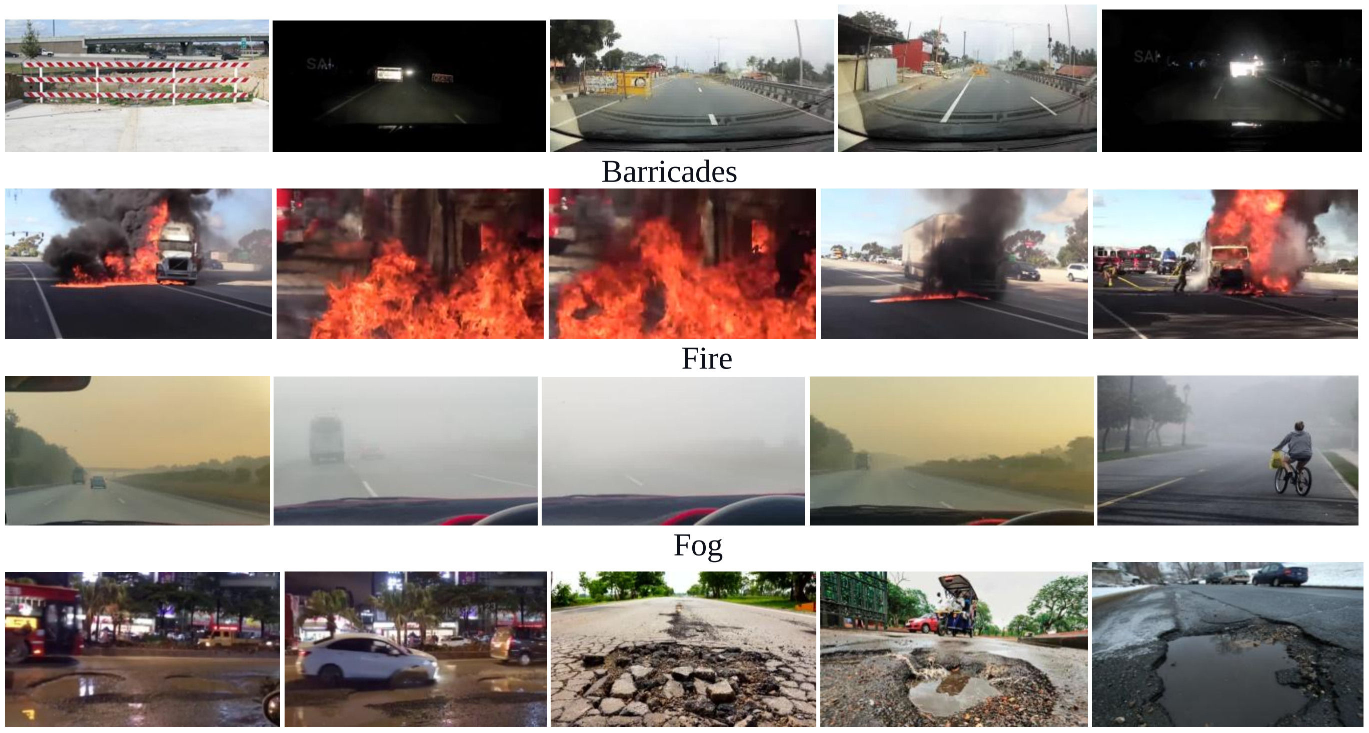


Fig 2.9: Road obstacles dataset: 4 classes: 300 images per class

3.3.3 Data Preprocessing:

Python OpenCV library is used to read the image into an array from the directory and it will also read the image as a grayscale image. After that image size is resized into 100x100. In the next step, the image array is declared as img_arr, and the label is declared for the target array. Here, img_arr will be the array of pixels and the label will be a value of 0 or 1. The next step is to append img_arr and label it into data. In the next, a loop is iterated over data and appended img_arr into a list x and label into a list y. The list x is a 4-dimensional array that consists of several images, the width of each image, the height of each image, and the number of channels. This list y indicates the number of images. Python sklearn library is used to split the dataset into an 80:20 ratio. Before the creation of the model, x_train and x_test are rescaled.

3.3.4 Model implementation:

A CNN model applies filters on the raw pixel of an image to learn the pattern by comparing the global pattern with an existing neural network. The model is implemented by applying the following steps

- Convolution layer: Apply 64 (3x3) filters (Extracting 3x3 pixel sub-regions) with RELU activation function.

- Pooling layer: Perform max pooling with a (2x2) filter and stride of 2.

- Convolution layer: Apply again 64 (3x3) filters with RELU activation function.

- Pooling layer: Perform again max pooling with a (2x2) filter.

- 1796 neurons with a dropping rate of 0.4.

- Fully connected layer: A fully connected layer is used to flatten the previous layers.

- Dense layer: Fully connected layer is followed by a dense layer with a RELU activation function.

- Output Layer: The final output layer of CNN consists of 4 layers for 4 classes(barricade, fog, fire, and potholes) with a softmax activation function.

Three important modules are used to implement a CNN

- conv2d() implements a two-dimensional convolution layer with several filters, padding, activation function, and kernel size as arguments.

- max_pooling2d() implements a two-dimensional pooling layer by max-pooling algorithm.

- dense() implements a dense layer with the hidden layers

Three parameters are considered when the model is compiled.

- Optimizer: Adam optimizer is used during the compilation of the model
- Loss Function: Categorical cross-entropy is used as a loss function.
- Metrics: Accuracy is used as a metric since it is a classification based problem

Model Summary

Table 1.2: CNN model summary

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 98, 98, 32)        896
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 47, 47, 64)        18496
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 23, 23, 64)        0
_________________________________________________________________
flatten_6 (Flatten)          (None, 33856)             0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               4333696
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_13 (Dense)             (None, 4)                 516
=================================================================
Total params: 4,353,604
Trainable params: 4,353,604
Non-trainable params: 0
_________________________________________________________________
```

3.3.5 Flowchart

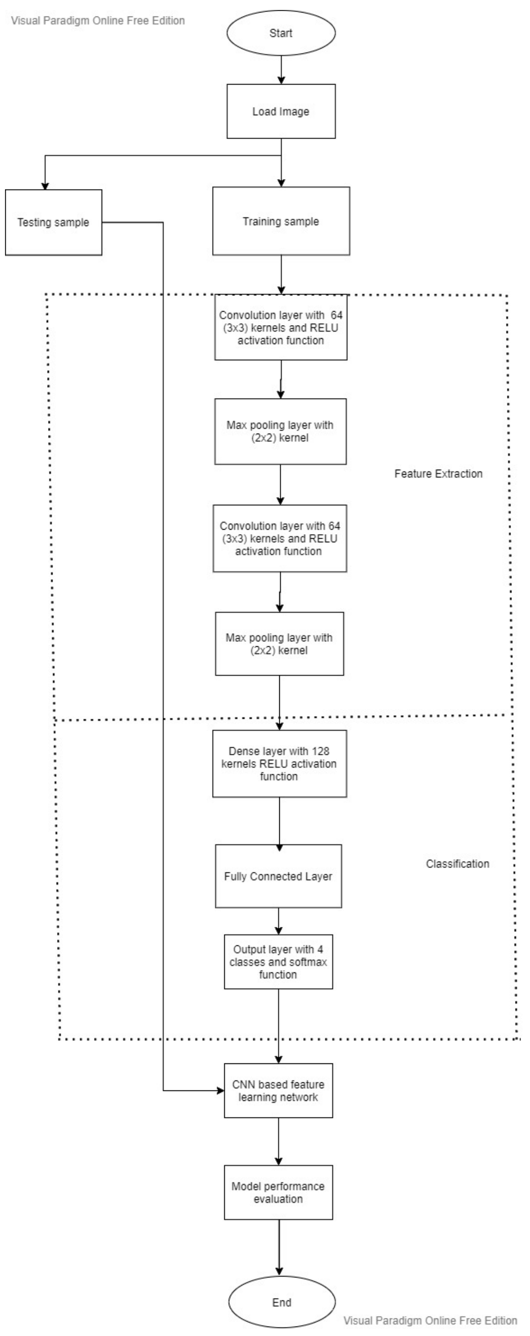

Visual Paradigm Online Free Edition
Start
Load Image
Testing sample
Training sample
Convolution layer with 64 (3x3) kernels and RELU activation function
Max pooling layer with (2x2) kernel
Feature Extraction
Convolution layer with 64 (3x3) kernels and RELU activation function
Max pooling layer with (2x2) kernel
Dense layer with 128 kernels RELU activation function
Fully Connected Layer
Classification
Output layer with 4 classes and softmax function
CNN based feature learning network
Model performance evaluation
End
Visual Paradigm Online Free Edition

Fig 3.0: Flowchart of CNN for obstacles classification

3.4 Environmental Sound detection

A sound is a form of energy that can be interpreted by compressions and rarefactions of the sound in the air. Sound is represented by a waveform that depicts the moving of air particles back and forth over time. The vertical axis represents the movement of sound either backward or forwards to time. The horizontal axis represents the time. A plot of sound is given below

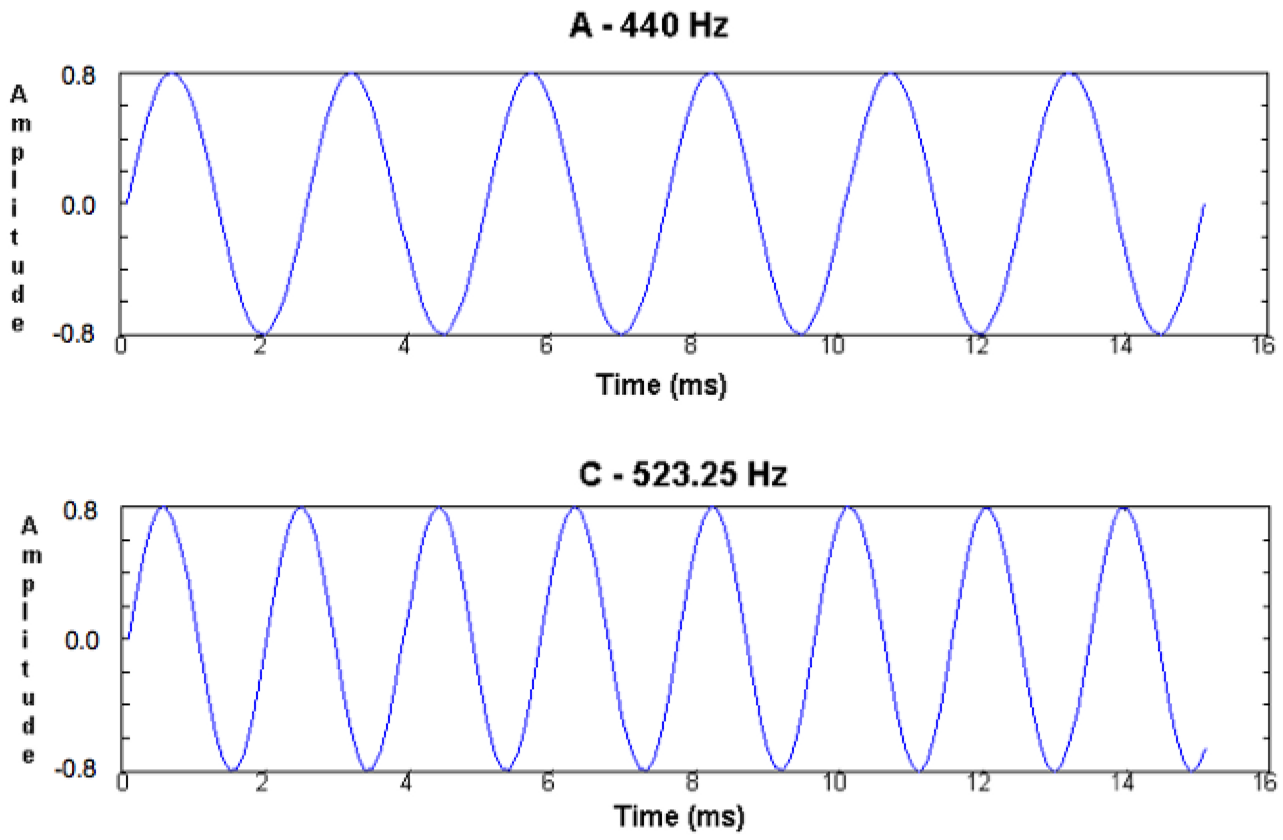


Fig 3.1: Wave form of sound shows two distinct points

In the above, points A and C are the pitch and their frequencies are 440 Hz and 523.25Hz.

3.4.1 Dataset:

To analyze, detect, and classify different environmental sounds. Urbansound8K is used. This dataset consists of 8732 labeled sound excerpts of Urbansound from 10 different classes. These classes are air_conditioner, car_horn, children_playing, dog_bark, engine_idiling, gun_shot, jackhammer, siren, street music.

The metadata has the following 8 columns

- Slice_file_name: name of the audio file.
- fsID: FreesoundID of the recording where the excerpt is taken from
- Start: start time of the slice.
- End: End time of the time.
- Salience: salience rating of the sound. 1=foreground, 2=background.
- Fold: The fold number(1-10) to which this file has been allocated

• classID:

0 = air_conditioner
1 = car_horn
2 = children_playing
3 = dog_bark
4 = drilling
5 = engine_idling
6 = gun_shot
7 = jackhammer
8 = siren
9 = street_music
•Class Name: class name

3.4.2 Data Exploratory

The Audio data present in the dataset are in .wav format. Sampling is used to digitize this sound wave at discrete intervals. These intervals are called the sampling rate. The typical sampling rate for these audio signals is considered 441.KHz. Each sample indicates the amplitude of the waveform at a particular interval, where bit depth indicates how the sample is referred to as the dynamic range of interval. For analysis of audio librosa library is used.

The count plot for all the classes is shown below

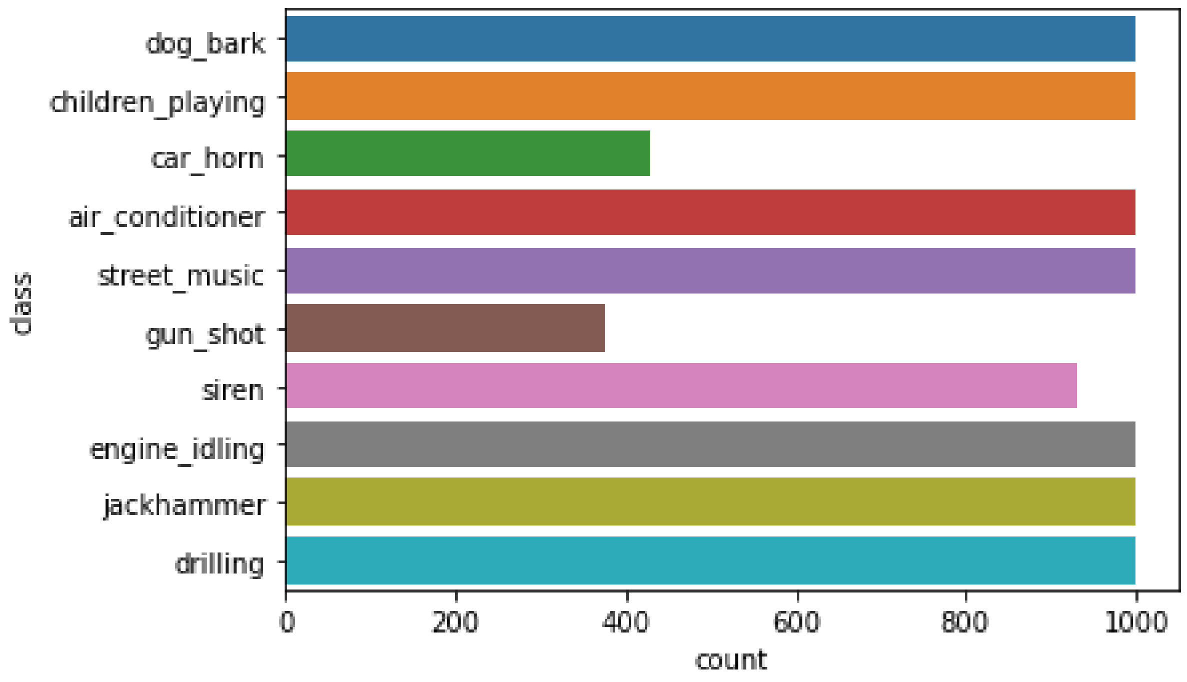


Fig 3.2: The countplot for each class label in the dataset

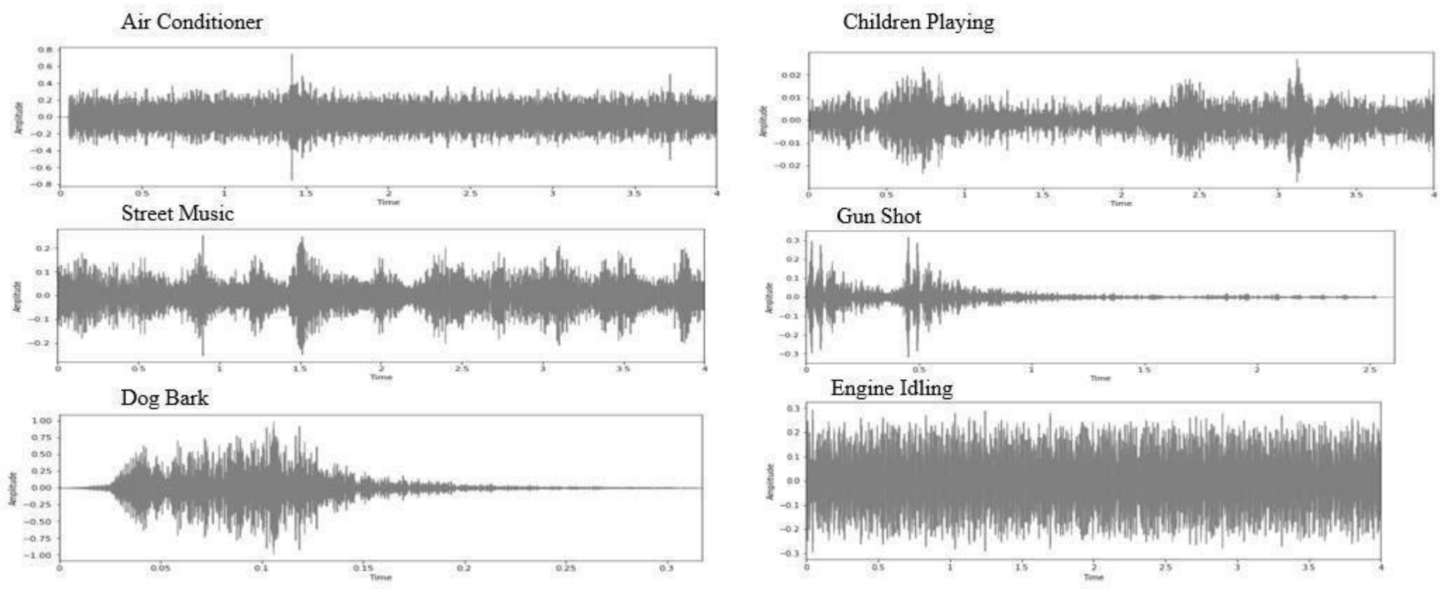


Fig 3.3: Each class label such as air condition. children playing. street music, gun shot, dog bark, engine idling waveform

Each audio file is in the format of .wav from which various formats can be extracted. Three major. properties are considered. Most samples have two audio channels (stereo) and some of them have one channel(mono). Various range of sample rate(96kHz to 8kHz) is used across all the samples. A wide range of bit rates (4bit to 32bit) is used.

3.4.3 Data preprocessing

Librosa’s load function() is used to preprocess the data. This function converts the sampling rate to 22.05kHz. and normalizes the data so the bitrate varied from -1 to 1 and flattens the stereo signal to mono signal.

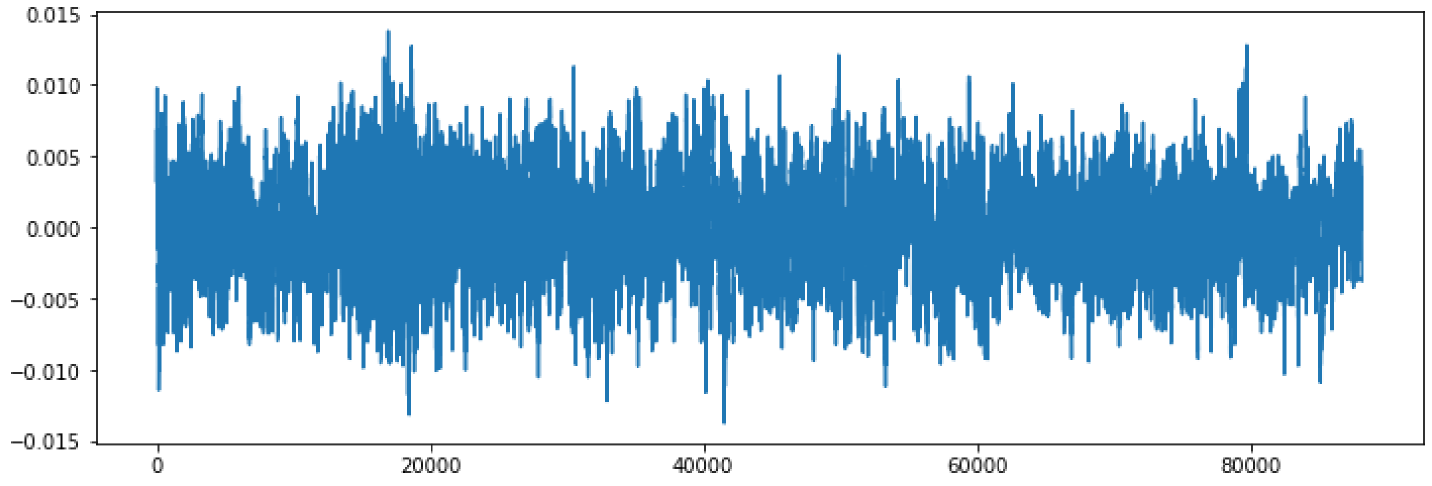


Fig 3.4: Single-channel signal(mono signal)

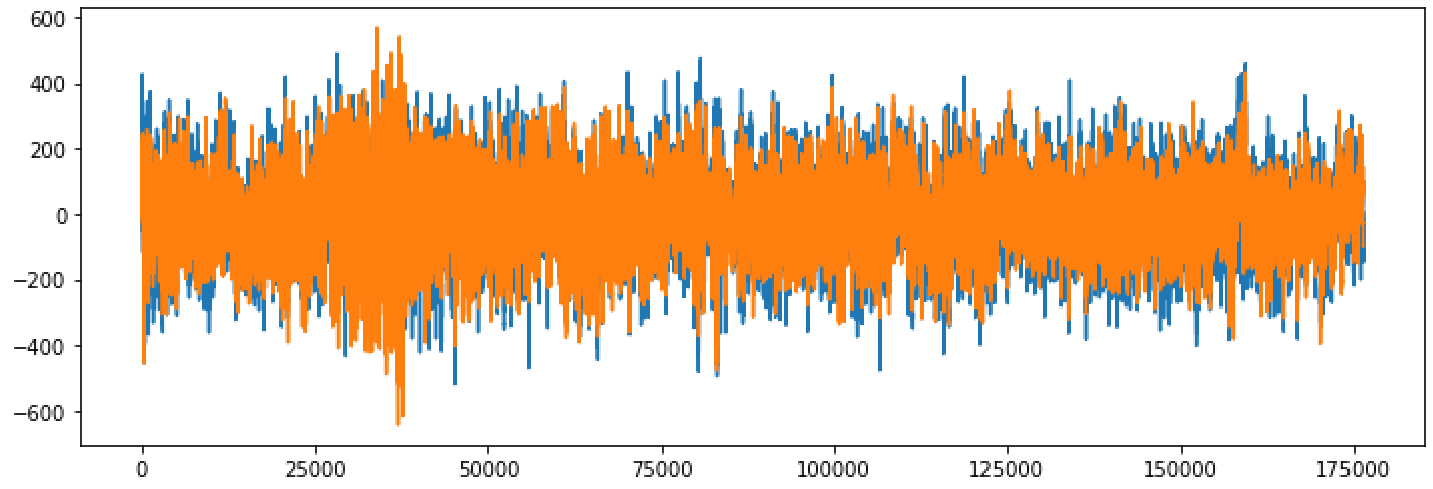


Fig 3.5: Double-channel signal(Stereo)

3.4.4 Feature Extraction:

The next step is to extract the essential feature to train the model. A visual representation that helps to track the feature for classification. The Mel-frequency cepstrum coefficient technique is used for visualizing the spectrum of the frequencies. An MFCC applies a quasi-logarithm spaced frequency scale for visualizing the spectrum.
In audio signal processing, the Mel-frequency cepstrum coefficient indicates the short-term power spectrum of a sound that can support a linear cosine transform of a log power spectrum on a non-linear Mel scale of frequency.

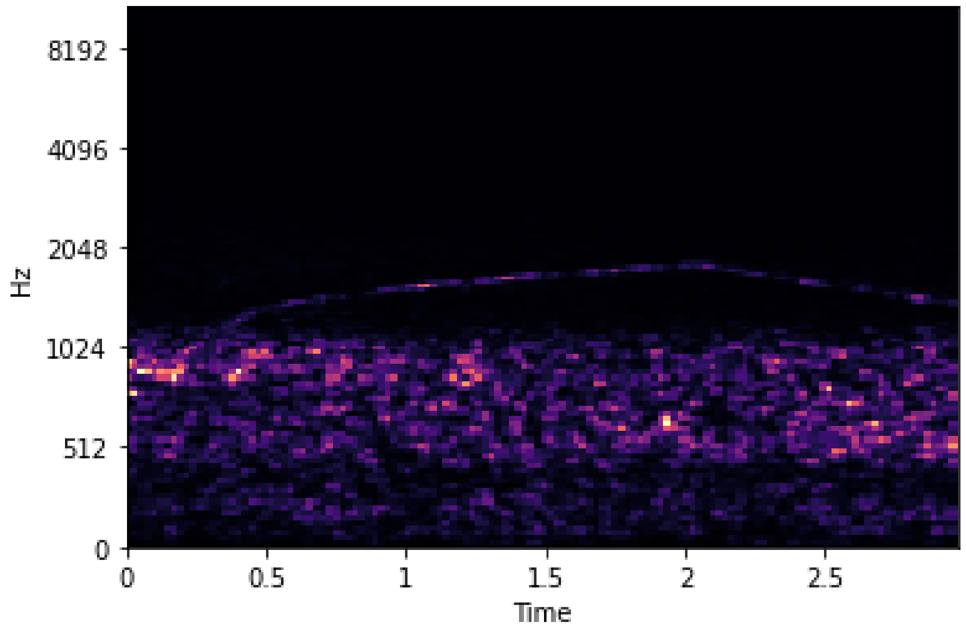


Fig 3.6: Frequency vs time waveform

In the sound dataset, an MFCC is extracted and stored in a pandas data frame along with the classification label. Librosa's mfcc() function is used for generating an MFCC from time series audio data.

3.4.5 Environmental sound detection and classification Using KNN

The proposed system uses K nearest algorithm to detect and classify the environmental sound. A detailed overview of the KNN algorithm is provided below

3.4.5.1Overview of KNN

K Nearest Neighbor is one of the crucial supervised machine learning algorithms which is applicable for regression and classification. The K in KNN represents the number of neighbors that classifies or predicts the outcome of the dataset. The prediction of every new data point is performed depending upon a particular distance measured from the nearest neighbor and weighted average.

3.4.5.2 Properties of KNN

- KNN is a lazy learning algorithm because it does not consider any particular training phase and takes all the data for training. KNN is used to find the relationship between each observation in a training phase.

- KNN is a non-parametric algorithm because it does not make any assumptions about the functional form of the relationship. It works directly on training samples rather than using any particular model.

- KNN considers a similarity between the new data point and available data point and places the new data point into the category which is almost equivalent to the available category.

- KNN stores the dataset at the training phase and when a new data point is obtained, the model will find the similarity between the new data point and the available categories.

3.4.5.3 Methods of calculating the distance for KNN

The distance of target data points can be calculated by measuring the interval between a new data point and each data point around it. The most common methods for measuring the distance are Euclidean distance, Manhattan distance, and Hamming Distance.

Euclidean Distance:

Euclidean distance is a straight line distance between two points on the Euclidean plane. This distance can be measured by calculating the square root of the sum of the squared difference between a new data point and an existing data point

If the new data point is x and the existing data point is y, then the Euclidean distance can be measured

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan Distance:

Manhattan distance is a real distance between the real vectors using the sum of their absolute difference.

The Manhattan distance can be calculated using the following formula

$$\sum_{i=1}^{k} |x_i - y_i|$$

Hamming Distance:

Hamming distance is used for categorical data. If the value of the given data point is equal to the point from which the data point is measured, then the Hamming distance is zero, otherwise it will be 1.

The Hamming distance can be calculated by using the following formula

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$
$$x \neq y \Rightarrow D = 1$$

3.4.5.4 How to select K value for KNN algorithm

The K value is an important parameter in the KNN algorithm and It is an important step to find the optimal value of K. The optimal value of K reduces the effect of the noise on the classification. The Elbow method helps to choose the optimal value of the K. Small values of K give a result with noise and a very large value of K is difficult to find. Therefore, the most preferred value for K is 6.

3.4.5.5 How K Nearest algorithm works for sound classification

The K nearest algorithm works in the following ways

- Selecting a value for K, Here, K represents the number of training.
- Calculating the distance of unknown data points from all the training.
- Searching for the K observations in the training data that are closest to the calculated unknown data point.
- Measuring the distance between the unknown data point and training data point.
- The training data which has the smallest value considered to be the nearest point.

3.4.5.6 Flowchart

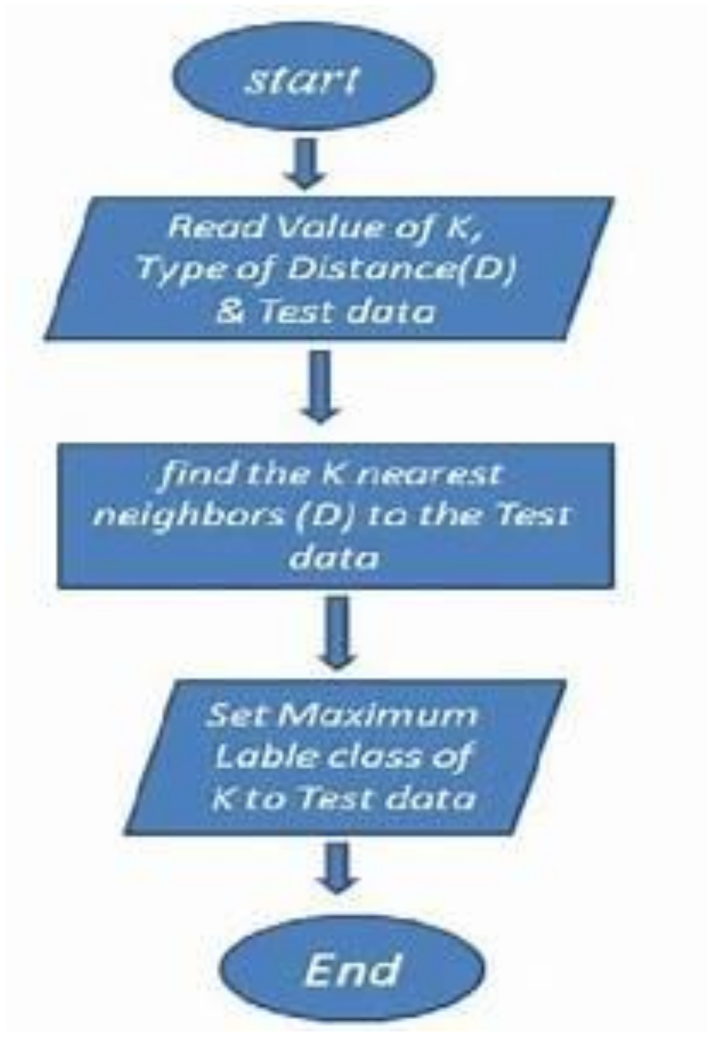

start
Read Value of K, Type of Distance(D) & Test data
find the K nearest neighbors (D) to the Test data
Set Maximum Lable class of K to Test data
End

Fig 3.7: Flowchart for sound classification using KNN

Chapter 4

Experimental Result

The study of self-driving car explores different functionalities of a car. To analysis the result multiple metrics are used and also these metrics are compared with result of existing researches.

4.1 Road lane detection using Line follower

To analysis the outcome of the line follower, a truth table is used. Here 0 is denoted by black and 1 is denoted by white, then the truth table for line follower will be

Table 1.3: Truth table for the output of the sensors

Left Sensor	Right Sensor	Output
0	0	Vehicle stops
0	1	Turn left
1	0	Turn right
1	1	Moves forward

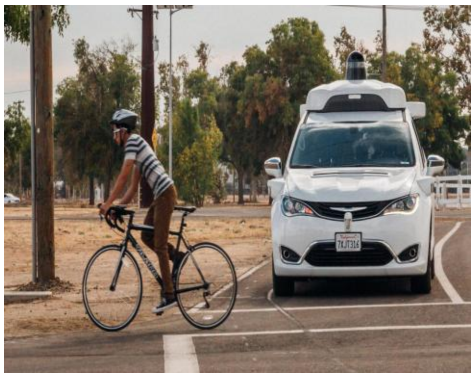
4.2. Object detection using YOLO

To analysis the performance of any object detector, mean average precision is considered. The map is used to make a comparison between the ground truth bounding box and detected box and generate a score. If the score is high, then the accuracy will be higher.

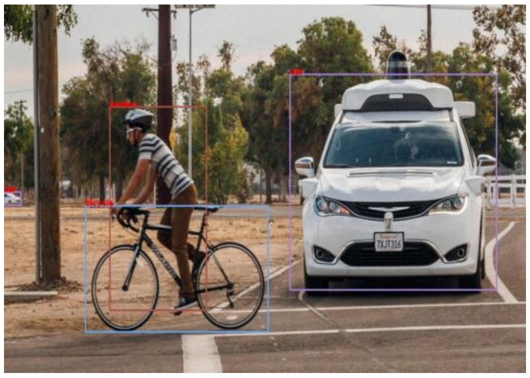
Mathematically. a Map can be written as

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} \mathrm{AveP(q)}}{Q}$$

4.2.1.Result of YOLO algorithm

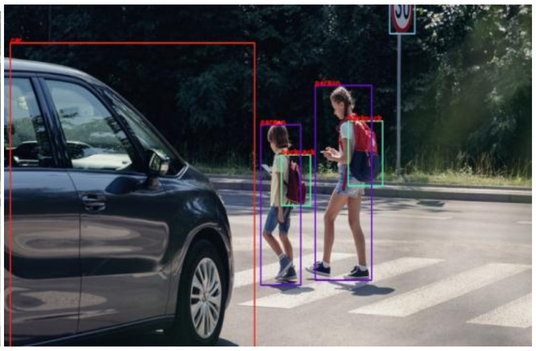


(i)



(ii)

(iii)


(iv)

(v)

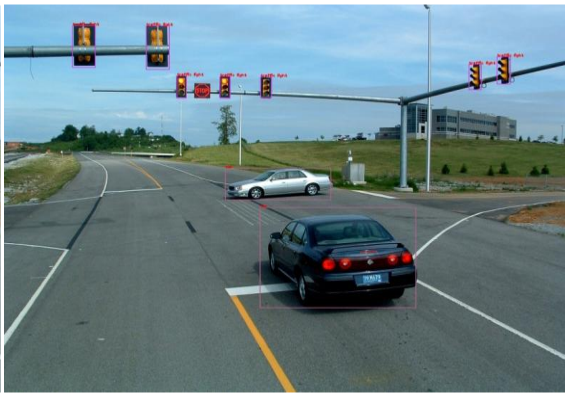
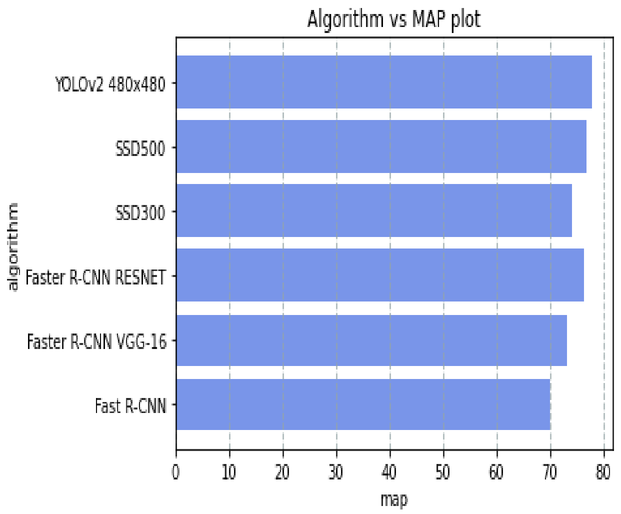
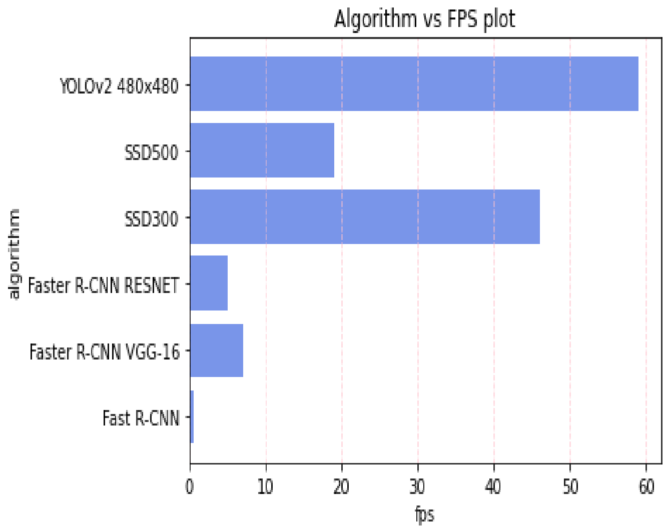

(vi)

Fig 3.8: (i),(iii),(v) the original picture and (ii),(iii),(vi) objects detected by YOLO algorithm

The mean precision score for the YOLO algorithm is 78.6 and FPS is 40. The comparative plot of map vs algorithms and comparative plot of fps vs algorithm are shown below



(a)



(b)

Fig 3.9: the image (a) shows the plot of algorithm vs MAP and (b) shows the plot of algorithm vs FPS.

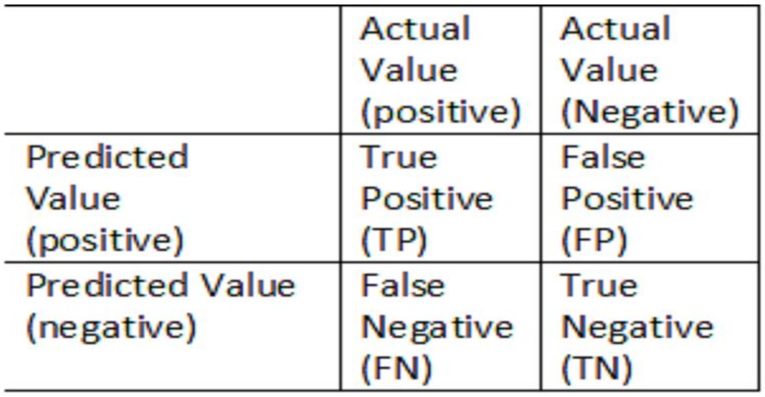
From the above diagram it is seen that YOLOv2 has generated more accuracy than other object detection algorithms and it is also cleared that the speed of the YOLO algorithm is high. It produces the highest Frame per second which helps to process a huge number of images.

Various parameters are defined to evaluate the performance of the classification model.

Confusion matrix:

A confusion matrix is a performance measurement tool that is used to evaluate the performance of classification-based problems. A confusion matrix is a form of a table that finds the performance of the classification model on a set of test data if the true value is known.

Confusion matrix can be 2x2 matrix, 3x3 matrix, or so on if the classifier of classes has 2 predictions, 3 predictions, or so on. The confusion matrix has two dimension-predicted value and actual value along with a total number of predictions. Actual values are the original values for the given observation and predicted values are the value predicted by the model.



	Actual Value (positive)	Actual Value (Negative)
Predicted Value (positive)	True Positive (TP)	False Positive (FP)
Predicted Value (negative)	False Negative (FN)	True Negative (TN)

Fig 4.0: abstraction of confusion matrix

True Positive: when the actual value and the predicted value both are yes.

False Positive: when the actual value is negative but the predicted value is positive.

True Negative: when the actual value and the predicted value both are no.

False Negative: when the actual value is positive but the actual value is negative.

Accuracy Score:

It is a performance evaluation metric which measures correctly identified cases.

Mathematically,

$$\text{Accuracy score}=\frac{Number\ of\ correct\ prediction}{Total\ number\ of\ prediction}$$

For classification based problem

$$\text{Accuracy score}= \frac{TP+TN}{TP+TN+FP+FN}$$

Classification report

A classical report evaluates the predictive power of a machine learning model. It measures the quality of predictions from the classification algorithm
It consists of the following parameters

- Precision
- Recall
- F1 score
- Support

Precision Score

It is a performance evaluation metric that determines the correctly identified positive cases from all the predicted positive cases. This is applicable when positive cases are high

Mathematically,

$$\text{Precision}=\frac{TP}{TP+FP}$$

Recall Score

It is a performance evaluation metric that determines the correctly identified positive cases from all the actual positive cases. This is applicable when negative cases are high.

Mathematically,

$$\text{Recall}= \frac{TP}{TP+FN}$$

F-1 Score

It is the weighted average or harmonic mean of precision and recall. It provides a better performance measure than the accuracy score.

Mathematically,

$$\text{F-1 Score}=2 * \frac{precision*recall}{precision+recall}$$

Support

It is the number of the actual existence of the class in a particular dataset

4.3Classification of road obstacles using CNN

To analyze the road obstacles using several parameters are considered.

The confusion matrix for this model is

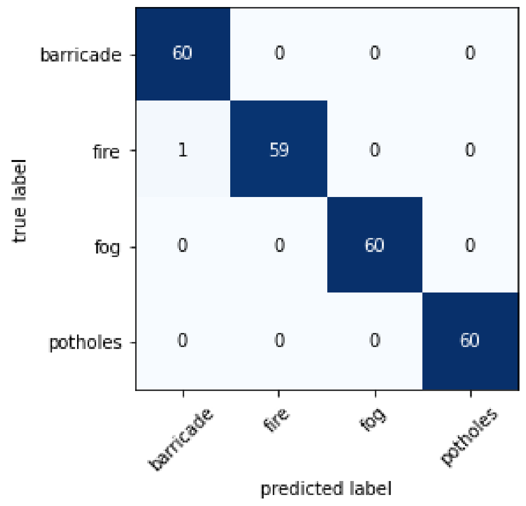


Fig 4.1: Confusion Matrix for obstacles classification using CNN

ii Accuracy score= 0.9958333333333333

iii. Classification report

Table 1.4: classification report for road obstacles

	Precision	recall	f1 score	support
barricade	0.98	1.00	0.99	60
Fire	1.00	0.98	0.99	60
Fog	1.00	1.00	1.00	60
potholes	1.00	1.00	1.00	60
Accuracy			1.00	240
macro avg	1.00	1.00	1.00	240
weighted avg	1.00	1.00	1.00	240

iv Precision score= 0.9958333333333333

v Recall score= 0.9958333333333333

vi. F-1 score= 0.9958333333333333

Vii Accuracy curve

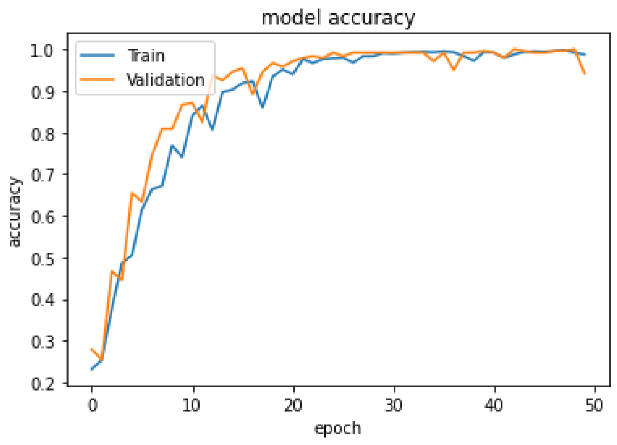


Fig 4.2: Accuracy vs Epoch curve for obstacles classification

vii. Loss curve

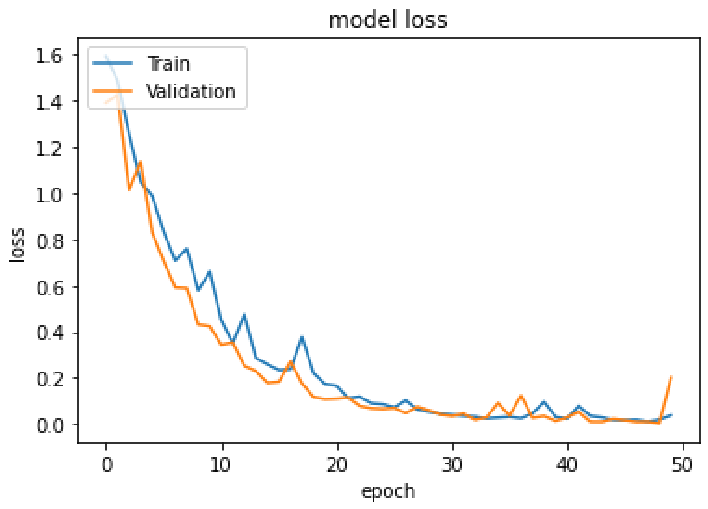
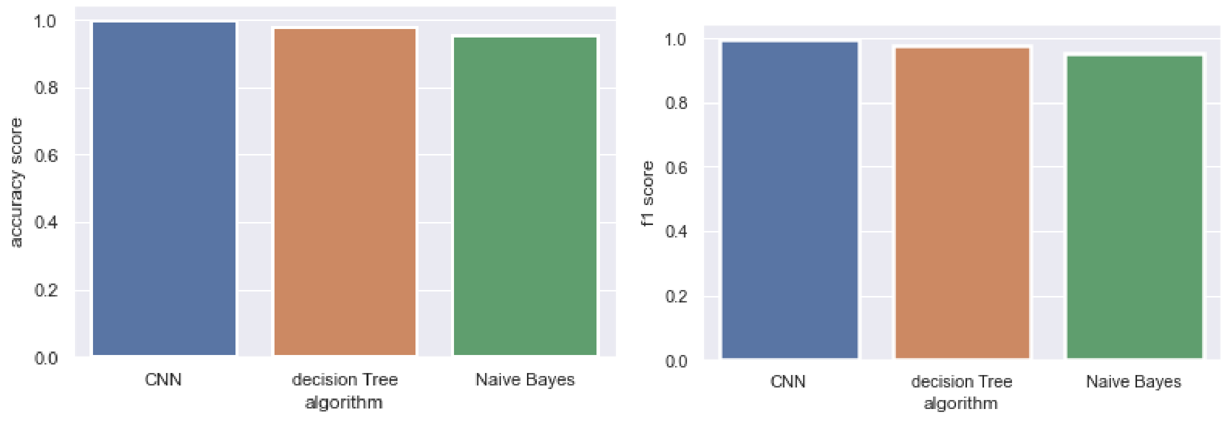


Fig 4.3: Loss vs Epoch curve for obstacles classfication

Performance comparison

The comparative study of different algorithms for the classification of road obstacles is shown below



(i) (ii)

Fig 4.4:performance comparison plot of different algorithms in terms of accuracy (i) and in terms of f1 score (ii)

4.4Urban sound classification using KNN

The following parameters are considered for analyzing the performance of the model i.confusion matrix

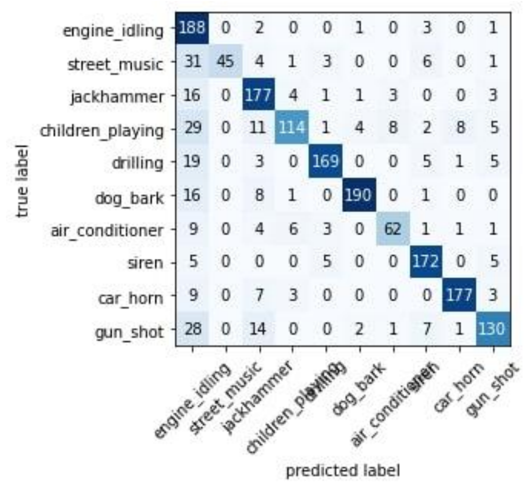


Fig 4.5: Confusion matrix for sound classification using KNN

ii.Accuracy score: 0.8151116199198626

iii.classificatio report

Table 1.5: Classificatio report for urban sound

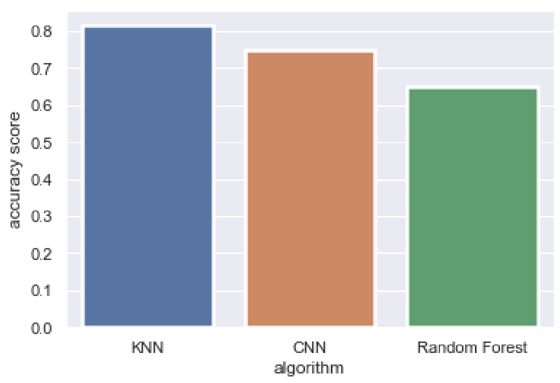
	precision	recall	f1 score	support
engine_idling	0.54	0.96	0.69	195
street_music	1.00	0.49	0.66	91
jackhammer	0.77	0.86	0.81	205
children_playing	0.88	0.63	0.73	182
Drilling	0.93	0.84	0.88	202
dog_bark	0.96	0.88	0.92	216
air_conditioner	0.84	0.71	0.77	87
Siren	0.87	0.92	0.90	187
car_horn	0.94	0.89	0.91	199
gun_shot	0.84	0.71	0.77	183
Accuracy			0.82	1747
micro average	0.86	0.79	0.80	1747
Weighted average	0.85	0.82	0.82	1747

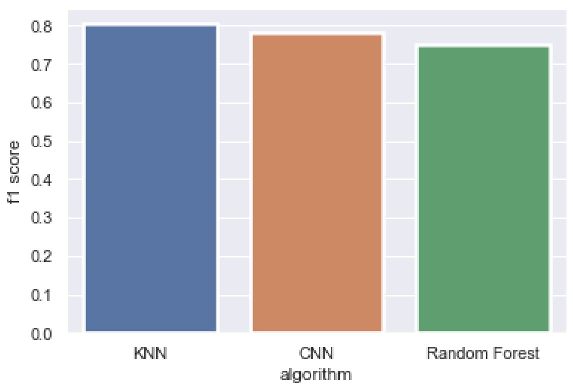
iv.Precision score: 0.8575175883330424

v.Recall score: 0.7896919136383704

vi. f 1 score: 0.8048929470764385

Performance comparison





(i) (ii)

Fig 4.6: Performance comparison of different models in terms of accuracy fig(i) and in terms of f1 score fig(ii)

Chapter 5

Conclusion and future work

5.1. Conclusion

In this project, different functionalities of a self-driving car have been discussed. Mainly four functionalities have been highlighted- road lane detection, surrounding object detection, classification of road obstacles, and environmental sound detection. A line follower approach is used to detect the road lane and this approach provides the optimum solution for road lane detection. You Only Look Once(YOLO) is applied to detect the objects around the car. In a study, it is observed that YOLO provides better performance than other detection approaches such as sliding windows, fast R-CNN, and SSD. Classification of road obstacles is performed using Convolution Neural Network and it is observed it provides the highest accuracy than other traditional machine learning approaches. Finally, K-Nearest- Neighbor is applied to classify the environmental sound which shows the highest accuracy.

5.2 Future Work

In this paper, some features of a self-driving car have been highlighted. Currently, the proposed system has four distinct features such as road lane detections, surrounding object detection, classification of road obstacles, detection, and classification of environmental sound. Still, some future works need to be done to build a fully equipped system. These works are road damage detection, optimum path selection, traffic light detection, and hoarding detection.

References

A Mandelbaum, D. W. (2017). *Distance-based confidence score for neural network classifiers*.

Abdulhakam.AM.Assidiq, O. O. (2008). *Real Time Lane Detection for Autonomous Vehicles.*

Brilian Tafjira Nugraha, S.-F. S. (2017). *Towards Self-driving Car Using Convolutional Neural Network and Road Lane Detector.*

Chan Yee Low, H. Z. (2014). Simple Robust Road Lane Detection Algorithm.

d (Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone. (2018).

Hyunggi Cho, Y.-W. S. (2014). A Multi-Sensor Fusion System for Moving Object Detection and Tracking in Urban Driving Environments.

Iurii Lezhenin, N. B. (2019). Urban Sound Classification using Long Short-Term Memory Neural Network.

Jonghee Sang, S. P. (2018). Convolutional Recurrent Neural Networks for Urban Sound Classification using Raw waveform.

Justin Salamon, J. P. (n.d.). UNSUPERVISED FEATURE LEARNING FOR URBAN SOUND CLASSIFICATION.

Pham, T. (2010). Non-maximum suppression using fewer than two comparisons per pixel.

Ping-Rong Chen, S.-Y. L.-M.-W.-J. (2018). *Efficient Road Lane Marking Detection with deep learning.*

Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone. (2018).

Ruturaj Kulkarni, S. D. (2018). Traffic Light Detection and Recognition for Self Driving Cars using Deep Learning.

SEM Bajestani, A. V. (2010). Technical report of building a line follower robot.

Simhambhatla, R. (2019). Self-Driving Cars: Evaluation of Deep Learning Techniques for Object Detection in Different Driving Conditions.

Thi Nhat Anh Nguyen, S. L. (n.d.). *Hybrid Deep Learning-Gaussian Process Network for Pedestrian Lane Detection in Unstructured Scenes.*

V Lempitsky, P. K. (2009). Image segmentation with a bounding box prior.

Walzel, M. H. (2018). Sensor and object recognition technologies for self-driving cars.