



***Dissertation on***  
**“Traffic Sign Board Recognition and Voice Caution Alert  
System Using CNN”**

*Submitted in partial fulfilment of the requirements for the award of degree of*

**Bachelor of Technology  
in  
Computer Science & Engineering**

**UE19CS390B – Capstone Project Phase - 2**

***Submitted by:***

<b>Abhinav Anand</b>	<b>PES1UG19CS016</b>
<b>Ambika Chapate</b>	<b>PES1UG19CS053</b>
<b>Devi Prasad D</b>	<b>PES1UG19CS138</b>
<b>Rudra Sharma</b>	<b>PES1UG19CS398</b>

*Under the guidance of*

**Prof. Supreetha S**  
Designation  
PES University

**August - December 2022**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
FACULTY OF ENGINEERING  
PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)  
100 Feet Ring Road, Bengaluru – 560 085, Karnataka, India



## **PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)  
100 Feet Ring Road, Bengaluru – 560 085, Karnataka, India

### **FACULTY OF ENGINEERING**

## **CERTIFICATE**

*This is to certify that the dissertation entitled*

**‘Traffic Sign Board Recognition and Voice Caution Alert System Using CNN’**

*is a bonafide work carried out by*

**Abhinav Anand  
Ambika Chapate  
Devi Prasad D  
Rudra Sharma**

**PES1UG19CS016  
PES1UG19CS053  
PES1UG19CS138  
PES1UG19CS398**

in partial fulfilment for the completion of seventh semester Capstone Project Phase - 2 (UE19CS390B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period August - December 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7<sup>th</sup> semester academic requirements in respect of project work.

Signature  
Prof. Supreetha S  
Assistant Professor

Signature  
Dr. Shylaja S.S.  
Chairperson

Signature  
Dr. B.K. Keshavan  
Dean of Faculty

### **External Viva**

**Name of the Examiners**

**Signature with Date**

1. \_\_\_\_\_

\_\_\_\_\_

2. \_\_\_\_\_

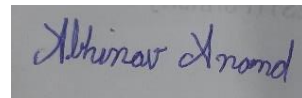
\_\_\_\_\_

## DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled “**Traffic Sign Board Recognition and Voice Caution Alert System Using CNN**” has been carried out by us under the guidance of Prof. Supreetha S, Assistant Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology in Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester August - December 2022. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

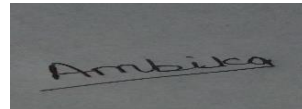
**PES1UG19CS016**

**Abhinav Anand**



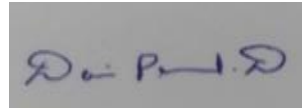
**PES1UG19CS053**

**Ambika Chapate**




**PES1UG19CS138**

**Devi Prasad D**



**PES1UG19CS398**

**Rudra Sharma**



## **ACKNOWLEDGEMENT**

We would like to express our gratitude to Prof. Supreetha S, Department of Computer Science and Engineering, PES University, for his/her continuous guidance, assistance, and encouragement throughout the development of this UE19CS390B - Capstone Project Phase – 2.

We are grateful to the project coordinator, Prof. Mahesh H.B., for organizing, managing, and helping with the entire process.

We take this opportunity to thank Dr. Shylaja S.S. Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support we have received from the department. We would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

We are deeply grateful to Dr. M.R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J., Vice-Chancellor, PES University for providing us various opportunities and enlightenment every step of the way.

Finally, this project could not have been completed without the continual support and encouragement we have received from our family and friends.

# **ABSTRACT**

Road signs are vital for ensuring a safe and smooth flow of traffic. Negligence in observing traffic signs and erroneously interpreting them is a major cause of road accidents. The suggested technology assists in identifying traffic signs and giving a voice notice to the driver via the speaker so that he or she may make the appropriate judgments. The suggested system is trained using a Convolutional Neural Network (CNN), which aids in the detection and categorization of traffic signs. To improve accuracy, a collection of classes is developed and trained on a specific dataset. The German Traffic Sign Benchmarks Dataset was utilised, which covers roughly 43 categories, 10 of which were chosen, 5 speed sign classes and 5 warning sign classes.

There are 19,957 photos of traffic signs, and the execution accuracy is almost 95%. Following the system's identification of the sign, a voice alarm is transmitted over the speaker to notify the driver. The proposed system also includes a segment in which the vehicle driver is informed to nearby traffic signs, allowing them to be aware of what laws to follow on the road. The goal of this system is to protect the driver, passengers, and pedestrians in the vehicle.

# TABLE OF CONTENTS

Chapter No.	Title	Page No.
1.	INTRODUCTION	01
2.	PROBLEM STATEMENT	02
3.	LITERATURE REVIEW	03
	3.1 Features recognition and random forests classification	
	3.1.1 Year	
	3.1.2 Authors	
	3.1.3 Methodology	
	3.1.4 Results	
	3.1.5 Future work	
	3.2 Recognition based on extreme learning machine	
	3.2.1 Year	
	3.2.2 Authors	
	3.2.3 Methodology	
	3.2.4 Results	
	3.2.5 Future Work	
	3.3 Classification using deep inception-based CNN	
	3.3.1 Year	
	3.3.2 Author	
	3.3.3 Methodology	
	3.3.4 Results and Findings	
	3.4 GTSRB	
	3.4.1 Year	
	3.4.2 Author	
	3.4.3 Summary	
4.	PROJECT REQUIREMENTS SPECIFICATION	07
5.	SYSTEM DESIGN	10
6.	PROPOSED METHODOLOGY	14
7.	IMPLEMENTATION AND PSEUDOCODE	23
8.	RESULTS AND DISCUSSION	29

<b>9. CONCLUSION AND FUTURE WORK</b>	<b>36</b>
<b>REFERENCES/BIBLIOGRAPHY</b>	<b>37</b>
<b>APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS</b>	<b>38</b>

# LIST OF FIGURES

<b>Figure No.</b>	<b>Title</b>	<b>Page No.</b>
<b>1</b>	<b>System Architecture</b>	<b>10</b>
<b>2</b>	<b>Use Case Diagram</b>	<b>10</b>
<b>3</b>	<b>Dataflow Diagram</b>	<b>11</b>
<b>4</b>	<b>Class Diagram</b>	<b>12</b>
<b>5</b>	<b>Activity Diagram</b>	<b>12</b>
<b>6</b>	<b>Sequence Diagram</b>	<b>13</b>
<b>7</b>	<b>Before Data Augmentation</b>	<b>15</b>
<b>8</b>	<b>After Data Augmentation</b>	<b>17</b>
<b>9</b>	<b>Example Image</b>	<b>19</b>
<b>10</b>	<b>Example Image – Edges</b>	<b>19</b>
<b>11</b>	<b>6x6 Matrix</b>	<b>19</b>
<b>12</b>	<b>Convolving 6x6 matrix using 3x3 filter</b>	<b>20</b>
<b>13</b>	<b>4x4 Matrix</b>	<b>20</b>
<b>14</b>	<b>After the Calculation</b>	<b>20</b>
<b>15</b>	<b>Resultant Matrix of 4x4 size</b>	<b>21</b>
<b>16</b>	<b>Command prompt running the app.py</b>	<b>29</b>
<b>17</b>	<b>First Opening Page of Web App</b>	<b>30</b>
<b>18</b>	<b>Login Page of Web App</b>	<b>30</b>
<b>19</b>	<b>Upload Page/Prediction Page</b>	<b>31</b>
<b>20</b>	<b>Prediction in Progress</b>	<b>31</b>
<b>21</b>	<b>Sign Predicted</b>	<b>32</b>
<b>22</b>	<b>Performance Analysis</b>	<b>32</b>
<b>23</b>	<b>Accuracy Chart</b>	<b>33</b>
<b>24</b>	<b>Accuracy Plot</b>	<b>33</b>
<b>25</b>	<b>Loss Plot</b>	<b>34</b>
<b>26</b>	<b>Jupyter Notebook Test Screenshot</b>	<b>34</b>



## **LIST OF TABLES**

<b>Table No.</b>	<b>Title</b>	<b>Page No.</b>
<b>1</b>	<b>Before Augmentation Frequency</b>	<b>17</b>
<b>2</b>	<b>After Augmentation Frequency</b>	<b>18</b>

# CHAPTER-1

## INTRODUCTION

Numerous technical developments have led to the development of vehicles having an autopilot mode. There are now autonomous automobiles on the road. The market for autonomous vehicles has seen tremendous growth. These amenities, however, are only found in a select few expensive vehicles that the public cannot purchase. To some extent, we sought to create a system that makes driving easier.

In India, there are accidents happening every day due to the drivers' ignorance of looking at the sign boards or because of the sign boards not being visible. This causes a huge problem and hence, there are systems that overcome this problem but at a very high cost.

Our project focuses on reducing the cost for the system and be readily deployable in every vehicle as a substitute to looking at the sign boards. A CNN model has been developed to recognise and classify the recognised images, hence making the drive easy. There is a huge boom in the development and production of smart cars in India. Traffic Sign recognition plays an important role in maintaining a city's traffic.

## CHAPTER-2

### PROBLEM STATEMENT

As we move on to the Fourth Industrial Revolution, the internet and the digital world mean real-time communication with additional components in the production line. The problem is that accidents occur no matter what revolution happens. One among them is the negligence of road signs and not following the traffic rules.

To make people follow the traffic rules, an interesting method of guiding the citizens is very much required, which is both accurate and efficient. Hence, we need to propose a method of recognising the traffic sign boards and notify the driver of the sign recognised. To do this, we also need to develop voice caution system.

Recognition systems based on Support Vector Machines have been proposed but accuracy issues persist. The infotainment system in the cars give us sophisticated information about the car and its current state, however displaying of the sign recognized isn't enough as most of the times the driver needs to focus on the road. To overcome this a voice alert as a caution needs to be implemented. Also, weather conditions depending on time of the day or night can cause a problem in identifying the signs and hence we need a better recognition system.

## CHAPTER-3

### LITERATURE SURVEY

#### 3.1 Traffic Sign Detection and Recognition Using Features Combination and Random Forests [4]

- a. Year – 2016
- b. Authors - Ayoub ELLAHYANI, Mohamed EL ANSARI, Ilyas EL JAAFARI
- c. Methodology - The three-step TSDR system, which is based on computer vision and developed for rapid, reliable traffic sign detection and identification, is described in this article. In the first step, the three elements of the Hue, Saturation, and Value (HSV) space are employed for picture enhancement and thresholding. Then, using the segmented images, they utilise the distance to border feature and the Random Forests classifier to detect circular, triangular, and rectangular forms. The final stage is to determine the information contained in the detected traffic signals. The four feature descriptors that we compare are Gabor, Local Binary Pattern (LBP), Histogram of Oriented Gradients (HOG), and Local Self-Similarity (LSS). They compare the numerous ways they may be combined. For the classifiers, they compared Random Forests with Support Vector Machines (SVMs). The best results were obtained by combining HOG with LSS and the Random Forest classifier. The Swedish Traffic Signs Data set was utilised to test the proposed technique, and the findings were positive.
- d. Results - This study proposes a rapid way for detecting and recognising traffic signs. In the first stage, we employ colour segmentation to narrow the search space. An enhancement and a thresholding were used on the HSV colour space. In the second step, the Distance to Border feature and a Random Forest classifier are used to detect

the circular, rectangular, and triangular signs. Using a mix of HOG and LSS features, the Random forests classifier is utilised to identify the observed candidates. The system achieves an accurate classification rate of more than 96% at a processing speed of 8-10 frames per second.

- e. Future work - In the future, adaptive thresholds might be utilised to overcome colour segmentation problems.

### **3.2 An Efficient Method for Traffic Sign Recognition Based on Extreme Learning Machine [5]**

- a. Year – 2017
- b. Authors - Zhiyong Huang, Yuanlong Yu, Jason Gu, and Huaping Liu
- c. Methodology - This study offers a computationally efficient traffic sign identification system (TSR). This suggested method is composed of two modules: 1) the histogram of the oriented gradient variation (HOGv) feature extraction, and 2) a single classifier trained using the extreme learning machine (ELM) method. The given HOGv feature provides a nice balance between redundancy and local features, allowing it to display odd forms more accurately. The classifier is a feedforward network with a single hidden layer. The random feature mapping is achieved by utilising the ELM approach to link the input and hidden layers, while only the weights between the hidden and output layers are trained. As a result, layer-by-layer fine-tuning is no longer necessary. Meanwhile, the cost function contains the output weights' norm.

3.2.1 Results - The ELM-based classifier can achieve an optimum and generalised solution for multiclass TSR. It can also strike a balance between recognition accuracy and computational expense. This suggested technique is tested on three datasets: the German TSR benchmark dataset, the Belgium traffic sign classification dataset, and the updated mapping and assessing the state of traffic infrastructure (revised MASTIF). The results of the experiments reveal that the

proposed technique achieves not only excellent recognition accuracy, but also extremely high computational efficiency in both the training and recognition accuracy, but also extremely high computational efficiency in both the training and recognition procedures in all three datasets.

- e. Future Work - In the future, this ELM-based classifier will be expanded for traffic sign identification and the number of hidden nodes will be learned.

### **3.3 Traffic Sign Classification Using Deep Inception Based Convolutional Networks [7]**

- a. Year - 2015
- b. Author – Mrinal Haloi
- c. Methodology – In comparison to previous techniques, the deep network for traffic sign classification proposed in this study outperforms others on the GTSRB. This deep network was built with the objective of concurrently capturing local and global data using a modified version of the inception module and spatial transformer layers. Adopting this feature allows our network to recognise intraclass data even when it is warped.
- d. Results and Findings - Because of the spatial transformer layer, this network is more robust to deformations such as translation, rotation, and scaling of the input pictures. Finally, they attain a 99.81% performance.

### **3.4 The German Traffic Sign Recognition Benchmark: A multi-class classification competition [8]**

- a. Year – 2011
- b. Author – J Stallkamp
- c. Summary - Advanced driver assistance systems must recognise traffic signs automatically, which is a challenging real-world computer vision and pattern recognition challenge. A massive realistic collection of over 50,000 pictures of traffic signs has been compiled. It displays how signs seem to the eye as a function of distance, illumination, ambient conditions, partial occlusions.

The sample has 43 classes with unequal class frequencies. Participants must categorise two exam sets, each with over 12,500 photographs. The results of the first of these sets, which was used in the first evaluation phase of the two-part challenge. The approaches utilised by the individuals who had the best results are briefly explained and contrasted with baseline performance and outcomes for human traffic sign identification. The results of the competition show that contemporary machine learning algorithms excel at the tough task of recognising traffic signs. On this dataset, participants performed well, attaining up to 98.98% correct recognition rate, which is equivalent to human performance. The way the test application was created resulted in some human error.

# CHAPTER-4

## PROJECT REQUIREMENT SPECIFICATION

### 4.1 Introduction

This part of the document contains the project plan for the development of “Traffic sign recognition and voice caution system using CNN”.

TSR (Traffic Sign Recognition) is a key component of Advanced Driver Assistance Systems (ADAS). The automated identification and recognition of traffic signs is critical to the management of the traffic-sign inventory. It provides an accurate and fast technique of handling traffic sign identification with the least amount of human labour. Most systems analyse traffic signs, however utilising the CNN model would produce more accuracy and be faster. It would be much easier to identify and tell the motorist if it were supplied as a voice alarm, making the trip easier.

### 4.2 Existing System

- Yadav et al. used the Support Vector Machine approach. The dataset was partitioned into 90/10 for training and testing purposes, and linear classification was used. Colour Segmentation, Shape Classification, and Recognition were used to get the required outcome.
- Anushree. A et al. utilised a Raspberry Pi to detect and recognise traffic signs with very little code. However, it requires the Raspberry Pi board for implementation, which is extremely expensive.



### **4.3 Disadvantages of existing system**

- One of the major barriers to applying deep learning to a wide collection of traffic-sign categories is a lack of a big dataset with several hundred different categories and a sufficient number of cases for each category.

### **4.4 Proposed System**

- We use Convolutional Neural Network to construct the Traffic Sign Board Recognition and Voice Alert System in our suggested system. Our system's ability to detect, recognise, and infer road traffic signs will be a huge aid to the driver.
- The automatic road signs identification system's goal is to recognise and categorise one or more road signs from live colour photos.
- We inform the motorist about the sign by utilising the voice of the detected sign board. The technology gives the driver real-time information from traffic signs, which includes the most vital and difficult jobs. Then, in advance of any hazard, provide an audio warning to the driver. This warning then allows the driver to make the necessary corrections in order to mitigate or fully prevent the occurrence.

### **4.5 Advantages of proposed system**

- The accuracy of the proposed system is 95% and this model turned out to give the best accuracy when tested with the GTSRB dataset.
- The detector performs pretty well even for some of the difficult cases.
- It is provided with a deep network for learning a large category as well.

## **4.6 System Requirements**

### **4.6.1 Hardware Requirements**

- |                   |   |                                     |
|-------------------|---|-------------------------------------|
| i. System         | : | Intel or AMD Processor (minimum i3) |
| ii. Hard Disk     | : | 500 GB.                             |
| iii. Monitor      | : | Any lcd or led display              |
| iv. Input Devices | : | Keyboard, Mouse                     |
| v. Ram            | : | 4 GB <                              |

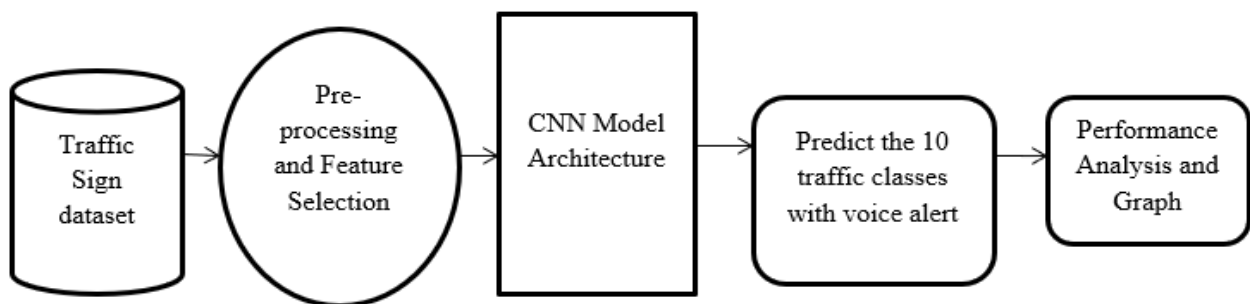
### **4.6.2 Software Requirements**

- |                     |   |                    |
|---------------------|---|--------------------|
| i. Operating system | : | Windows 7 or above |
| ii. Coding Language | : | Python3            |
| iii. Web Framework  | : | Flask              |

## CHAPTER-5

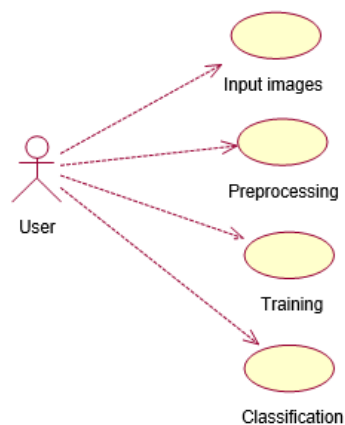
# SYSTEM DESIGN

### 8.1 System Architecture



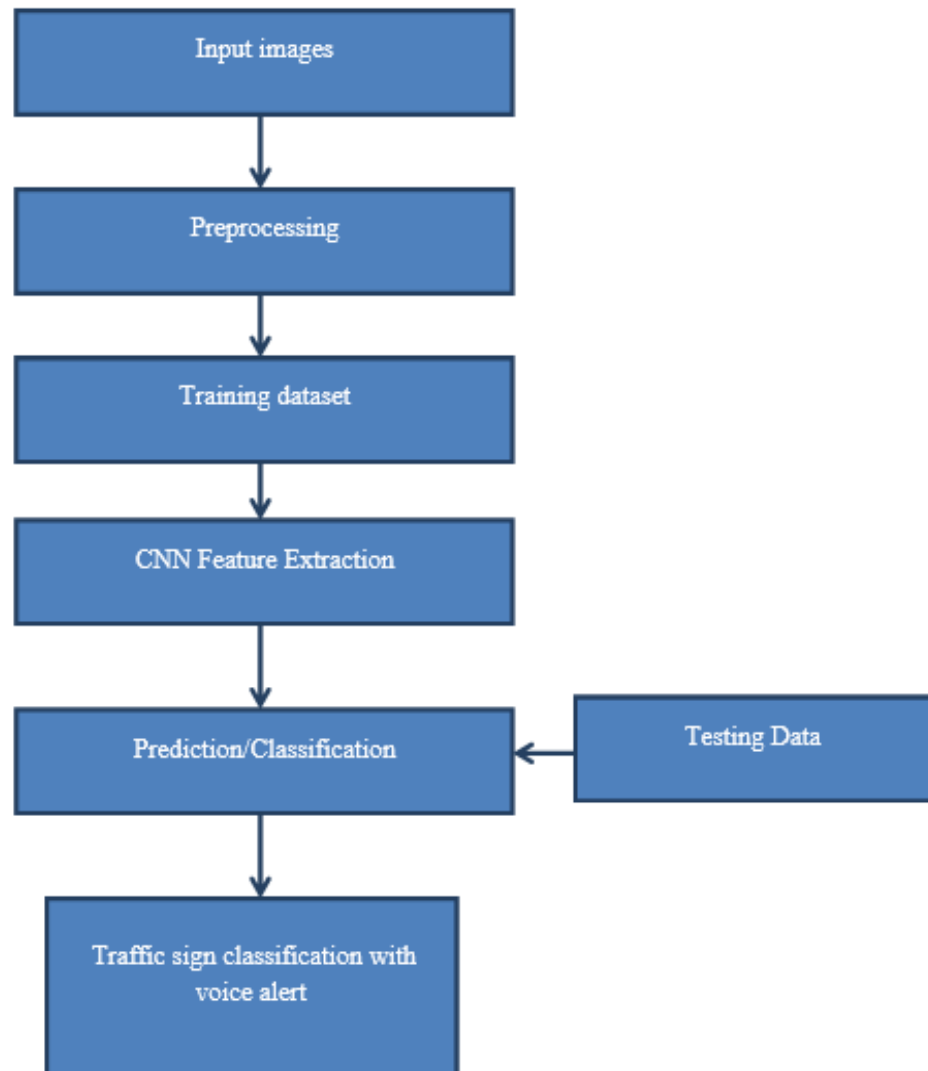
*Figure 1. System Architecture*

### 8.2 Use Case Diagram



*Figure 2. Use Case Diagram*

### 5.3 Dataflow Diagram



*Figure 3. Dataflow diagram*

## 5.4 Class Diagram

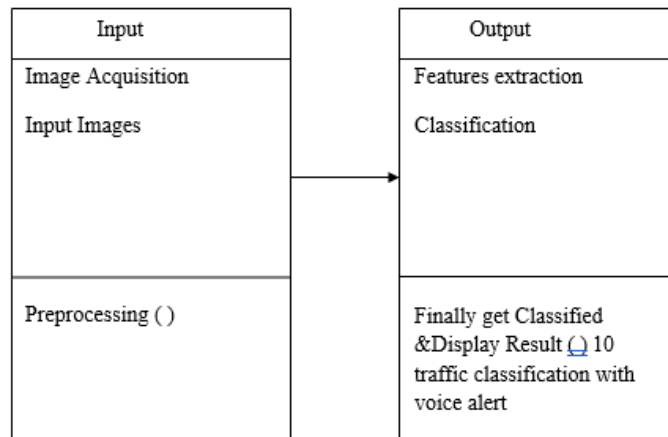


Figure 4. Class Diagram

## 5.5 Activity Diagram

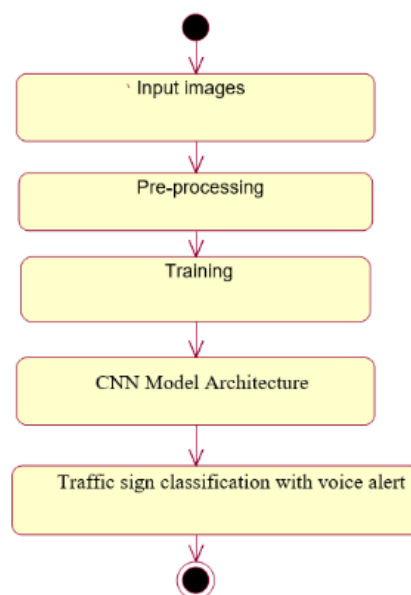


Figure 5. Activity Diagram

## 5.6 Sequence Diagram

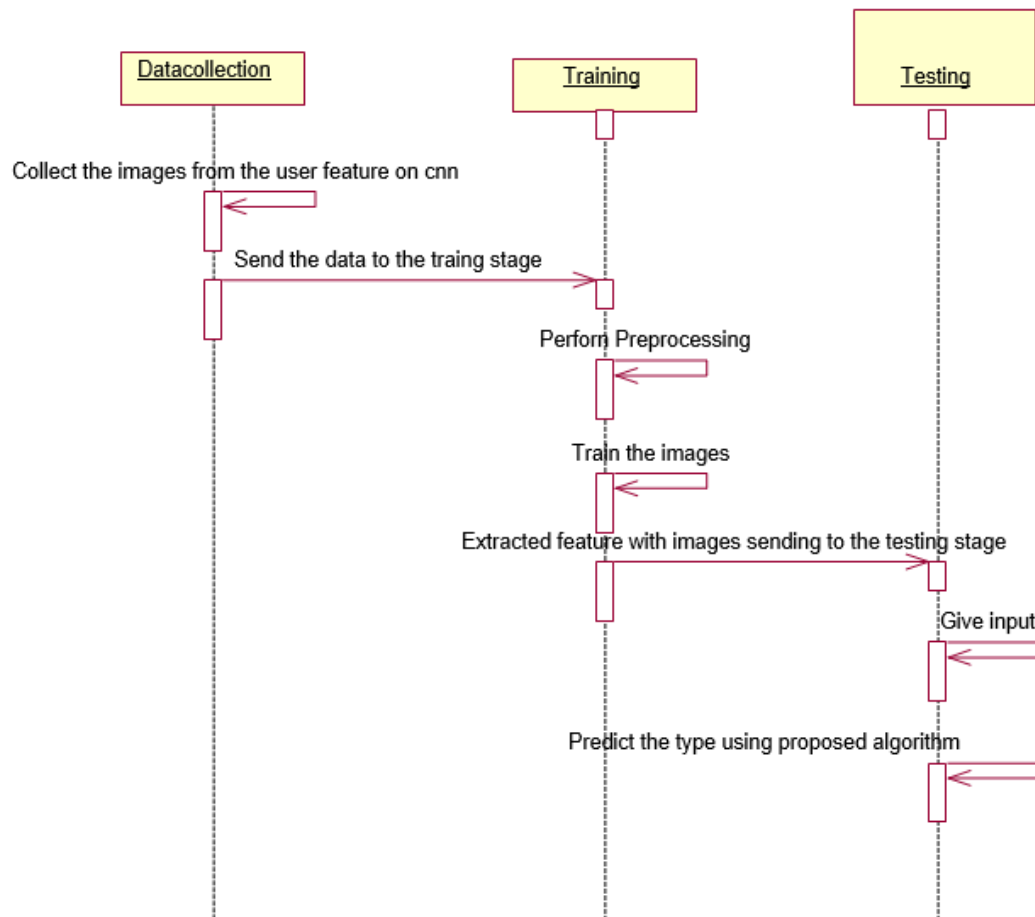


Figure 6. Sequence Diagram

# CHAPTER-6

## PROPOSED METHODOLOGY

### 6.1 Overview

The first step in designing is to get the data from the sources and based on this obtained dataset we need to build a model using any suitable architecture to obtain a model that can be used in recognising the traffic signs accurately. After obtaining the desired accuracy, we have to build a web application which plays the role of a car taking up images and hence feeding it to the model which in turn provides us with the voice caution and acts as an alert to the driver.

### 6.2 Research Design and Methods

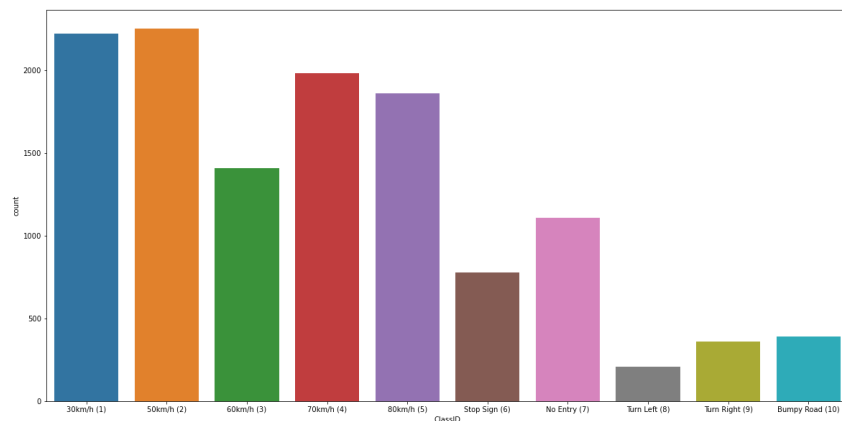
#### 6.2.1 Dataset Preparation

We need to prepare the dataset by ourselves or by obtaining a dataset that is already present on the internet that is collected by somebody else. Here, we have chosen the GTSRB (German Traffic Sign Recognition Benchmark) dataset from Kaggle.com. This dataset is about 600 mb and consists of 50,000+ images in total and consists of 43 classes of German traffic signs. Under which 10 classes have been further shortlisted for our project and they are 5 classes under the category of speed signs, i.e., 30km/h, 50km/h, 60km/h, 70km/h, 80km/h and 5 classes under the category of cautionary signs, i.e., Stop Sign, No Entry Sign, Turn Left Sign, Turn Right Sign and bumpy road signs.

### 6.2.2 Dataset Augmentation

Data Augmentation refers to artificial generation of data in order suffice the needs of training the model to get a higher accuracy. It involves random generation of data, be it image data or any other data.

In the 10 classes that we have chosen, the total number of images are 12,570 and data in each class is unbalanced w.r.t the other classes. This leads to underfitting of some classes and overfitting of some classes. Following is the frequency of each class.



*Figure 7. Before Data Augmentation*

From the above chart, we can see the data is unbalanced in the cautionary sign category while only one class in the speed sign category is unbalanced. To overcome this, we have used 2 different tools to augment the image data, ‘Augmentor’ and ‘ImageDataGenerator’. Augmentor is used for artificial generation of image data and is a python package. Imagedatagenrator is module under the Keras library used to generate batches of images for machine learning purposes.



1. Augmentor -

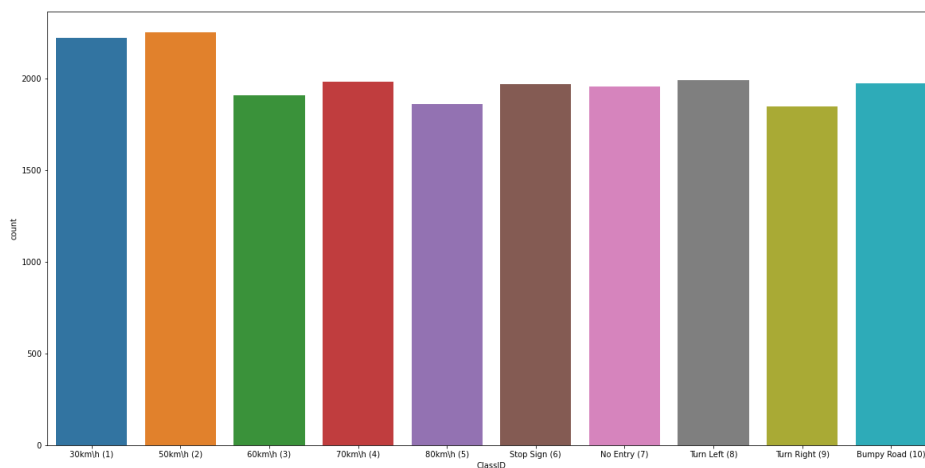
- a. Features Used – Zoom, Random Brightness, Random Distortion
- b. Augmentor uses PIL library to output the augmented images.
- c. Used Augmentor to –
  - i. 60 Speed Sign board
  - ii. No Entry Sign board
  - iii. Turn Left Sign board
  - iv. Turn Right Sign board
  - v. Bumpy Road Sign board

2. Image Data Generator –

- a. A module under preprocessing of Keras API, used to generate batches of image data.
- b. We create an object with the required arguments as the following –
  - i. Rotation\_range – rotates the image considering the current position as zero degrees with respect to the axis.
  - ii. Width\_shift\_range – Given image is shifted width wise.
  - iii. Height\_shift\_range – Given image is shifted height wise.
  - iv. Shear\_range – specifies the angle of slant.
  - v. ZCA\_whitening – If set to True, whitens the image. (Zero phase component Analysis)
  - vi. Zoom\_range – Zooms the image keeping the current scale to 0.
  - vii. Fill\_mode – If set to ‘nearest’, fills the points outside the boundaries.
  - viii. Brightness\_range – Specifies the brightness that each image must have in the specifies range.

c. Used to augment –

- i. Stop Sign
- ii. No Entry Sign
- iii. Turn Left Sign
- iv. Turn Right Sign
- v. Bumpy Road



*Figure 8. After Data Augmentation*

Signboard	Frequency
30 km/h	2220
50 km/h	2250
60 km/h	1410
70 km/h	1980
80 km/h	1860
Stop	780
No Entry	1110
Turn Left	210
Turn Right	360
Bumpy Road	390

*Table 1. Before Augmentation Frequency*

Signboard	Frequency
30 km/h	2220
50 km/h	2250
60 km/h	1910
70 km/h	1980
80 km/h	1860
Stop	1969
No Entry	1956
Turn Left	1992
Turn Right	1846
Bumpy Road	1974

Table 2. After Augmentation Frequency

### 6.2.3 Retrieving Images

We will retrieve the images and their labels. The photos should then be resized to (30,30), as all photographs should be the same size for recognition. The photos are then converted into a NumPy array.

### 6.2.4 Splitting Dataset into Train and Test Sets

Divide the dataset into two parts: train and test. There is 75% train data and 25% test data.

### 6.2.5 Building the model

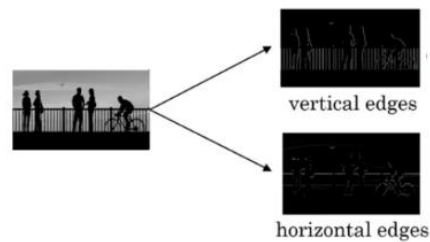
1. Convolutional Neural Networks –
  - a. One key issue with computer vision difficulties is that the input data can become quite large. Assume a picture with dimensions of 68 X 68 X 3. The input feature dimension is thus 12,288.

- a. This will become much more significant if we have larger photos (say, of size 720 X 720 X 3). If we feed such a large amount of data into a neural network, the number of parameters will skyrocket (depending on the number of hidden layers and hidden units). More processing and memory needs will follow, which most of us cannot handle.
- b. Edge detection example - the first layers of a neural network identify picture edges. Deeper levels may be able to identify the cause of the items, and even deeper layers may be able to detect the cause of entire things. Suppose we are given the below image.



*Figure 9. Example Image*

- c. The picture has a lot of vertical and horizontal edges. The first step is to identify these edges.



*Figure 10. Example Image – Edges*

- d. Create a 6-by-6 grayscale picture (with only one channel).

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*Figure 11. 6x6 Matrix*

- e. Use a 3 X 3 filter to convolve this 6 X 6 matrix.

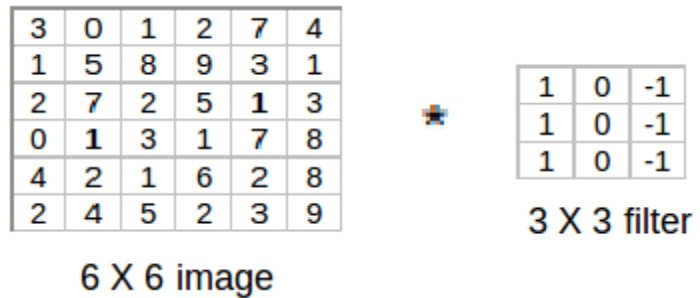


Figure 12. Convolving 6x6 matrix using 3x3 filter

- f. A 4 X 4 picture will be produced. The first member of the 4 X 4 matrix will be computed as follows.

3 <sup>1</sup>	0 <sup>0</sup>	1 <sup>-1</sup>
1 <sup>1</sup>	5 <sup>0</sup>	8 <sup>-1</sup>
2 <sup>1</sup>	7 <sup>0</sup>	2 <sup>-1</sup>

Figure 13. 4x4 Matrix

- g. We multiply the filter with the first 3 X 3 matrix from the 6 X 6 picture. The first element of the 4 X 4 output will now be the total of the element-wise products of these values, which is  $3*1 + 0 + 1*-1 + 1*1 + 5*0 + 8*-1 + 2*1 + 7*0 + 2*-1 = -5$ . To compute the second element of the 4 X 4 output, we will move our filter one step to the right and calculate the sum of the element-wise product once more.

0 <sup>1</sup>	1 <sup>0</sup>	2 <sup>-1</sup>
5 <sup>1</sup>	8 <sup>0</sup>	9 <sup>-1</sup>
7 <sup>1</sup>	2 <sup>0</sup>	5 <sup>-1</sup>

Figure 14. After the calculation

- h. We will convolve the entire image to obtain a 4 X 4 result.

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Figure 15. Resultant matrix of 4x4 size

- i. Convolution of a 6 X 6 input with a 3 X 3 filter produced a 4 X 4 output.
- j. In a CNN, there are basically three layers:
- i. Convolution layer
  - ii. Pooling layer
  - iii. Fully connected layer
- k. Pooling layers are commonly used to minimise the number of inputs and hence speed up computations.
2. Building our model – We will utilise the sequential model from the Keras library to create the model. The layers will then be added to create a convolutional neural network. We utilised 32 filters in the first two Conv2D layers, and the kernel size is (5,5). We retained pool size (2,2) in the MaxPool2D layer, which means it will choose the maximum value of every 2 x 2 area of the picture. By doing so, the image's size will be reduced by a factor of two. In the dropout layer, we kept the dropout rate at 0.25, which indicates that 25% of neurons are deleted at random. We repeat these three layers with a few tweaks to the settings. The flatten layer is then used to transform 2-D data to 1-D vector. This layer is followed by a dense layer, a dropout layer, and another dense layer. The last dense layer produces 9 nodes since our dataset divides traffic signs into 10 categories. This layer employs the SoftMax activation function, which assigns a probability value to each of the ten alternatives and forecasts which has the highest likelihood.

### **6.2.6 Applying model and plot graphs for accuracy and loss**

We will build the model and use the fit function to apply it. There will be 32 batches. The graphs for accuracy and loss will then be plotted. We obtained a training accuracy of 95% on average.

### **6.2.7 Saving the Model**

The following step is to save it as a.h5 or.pkl file.

### **6.2.8 Integrating Model with Web APP**

'pyttsx3' is a Python text-to-speech conversion package. It operates offline, unlike other libraries, and is compatible with Python 2 and 3. To obtain a reference to a pyttsx3. Engine instance, an application calls the pyttsx3.init() factory method. It is a simple tool that turns the entered text into voice.

The pyttsx3 module supports two voices, one female and one male, which are provided by "sapi5" for Windows.

# CHAPTER-7

## IMPLEMENTATION AND PSEUDO CODE

### 7.1 Implementation

1. Pre-processing the images –
  - a. Create lists – Data and Labels
  - b. Get current path
  - c. Append images on to an array, after resizing.
2. Converting lists to NumPy arrays –
  - a. Using `np.array()`
  - b. Saving the arrays for future use, using `np.save()`
3. Splitting the dataset –
  - a. Using scikit's 'train\_test\_split' under `sklearn.model_selection`.
  - b. Input–
    - Arrays=Data and Labels
    - Test size = Represents the proportion of dataset to include in the test split. (0.2 – value used)
    - Random State = Controls the shuffling before applying split. (0 – value used, integer value for an output)
4. Printing the shape of train and test –
  - a. Using NumPy's `array.shape()` – we get lengths of the split train and test data in the form of tuple.



5. Converting labels to one hot encoding-
  - a. Uses Keras Util's 'to\_categorical()' which takes an input of array and converts different categories to an array or matrix that has binary values.
  - b. Used to improve predictions as well as classification accuracy of model.
6. Building the model –
  - a. Defining the model layers –
    1. Sequential model is used appropriately for a plain stack of layers, having one input and one output.
    2. A total of 12 layers (6 when grouped) are created.
    3. Conv2D creates a kernel, which helps produce tensor outputs.
    4. Kernel is convolutional matrix used in blurring, sharpening, edge detection by doing a convolution between a kernel and an image.
    5. 'filters' is a mandatory parameter. Here, we have used 32 filters for first 2 layers.
    6. 'kernel\_size' refers to the dimensions of kernel and is set to 5x5 for first 2 layers. Using 5x5 to learn large features and then quickly reduced to 3x3.
    7. 'activation' used here is 'rectified linear unit', which for every positive value of x gives a positive value of y, equal to x.
    8. 'maxpool2d' is used to get a feature map containing most prominent features. Set to a pool window size of 2x2.
    9. 'Dropout' rate is set to 0.25, says that 25% of incoming neurons are nullified. Helps in regularization.
    10. Another 2 layers of conv2d with 64 filters and kernel size of 3x3 along with 'relu' activation function are created, followed by a layer of maxpooling of pool size 2x2 and a dropout layer of rate 0.25.
    11. 'flatten ()' layer is created to flatten the image dimensions after convolution of images.

12. 'activation' used here is 'rectified linear unit', which for every positive
13. A 'dense' layer is created to form a fully connected layer.
14. Another 'dropout' layer is created now with a rate of 0.5
15. Lastly a 'dense' layer with the 'softmax' activation function is created.
- b. Configuration of model –
  1. Model.compile is a method which does configuration.
  2. Has params, 'loss', 'optimizer', 'metrics'.
  3. Categorical\_crossentropy is a loss function for multi class classification.
  4. Optimizer used is 'adam'
- c. Training the model –
  1. Model.fit is used to train the model by passing the required data such as train data and test data for validation, along with batch\_size specified and number of epochs specified.
7. Plotting Accuracy and Loss Graphs –
  - a. To plot the accuracy and loss graphs, matplotlib which is a plotting library in python, is used.
  - b. The graphs plotted are – (i) Accuracy v/s Epochs (ii) Loss v/s Epochs
  - c. Both training and validation metrics are shown.
8. Saving the trained model –
  - a. Model.save is used to save the trained model.
9. Implementing the web application –
  - a. Web framework used – Flask
  - b. Libraries used – werkzeug, keras, numpy, PIL, pyttsx3
  - c. Werkzeug.utils, under which secure\_filename is used, to which passing a filename, returns a secure version of it.
  - d. Keras.models, under which load\_model is used to load the model onto the web application.
  - e. Numpy to append image on to the array.
  - f. PIL to grab images.
  - g. 'app.py' consists of loading and predicting the model.

- h. We are greeted with the login page where we need to login using the credentials.
- i. Next, we are open to upload any image and clicking on predict gives us the voice message of the recognized traffic sign.
- j. We can check for the accuracy plot and also the metrics by using the appropriate options on the top right.

## **7.2 Pseudo Code for model building**

Initialize

```
Install > opencv  
Install > tensorflow  
Install > keras
```

```
Import > numpy, pandas, PIL
```

Start process

```
#Appending image data
```

```
Create array > data [], labels []  
Define classes = No. of classes of signs  
  
for (go through classes)  
    for (go through images in each class)  
        resize images  
        append array(images) onto data ()
```

```
#Converting Lists to Numpy arrays
```

```
np.array > data  
np.array > labels
```

```
#Loading data and Labels
```

```
np.load > data  
np.load > labels
```

```
#Splitting for train and test
```

```
Import > train_test_split, to_categorical
```

```
Train_test_split (input = data, labels; size_of_test = 0.2)

#Converting labels to one hot encoding

y_train > to_categorical(y_train, No. of classes)
y_test > to_categorical(y_test, No. of classes)

#Model

Import > tensorflow, Sequentail, Conv2D, Maxpool2D, Dense, Flatten,
Dropout

Model = Sequentail model
Add 2 Conv2D layers with filters = 32 kernel size = 5,5 with activation =
'rectified liner activation'.
Add a layer of maxpool2d with pool size = 2,2

Add a dropout layer with 25% of rate of dropping.

Add 2 Conv2D layers with filters = 64, kernel size = 3,3 with activation =
'rectified linear activation'.
Add a layer of maxpool2d with pool size = 2,2

Add a dropout layer with 25% of rate of dropping

Flatten () the model

Add a dense layer with 256 neurons input with 'rectified linear activation'
function.

Add a dropout layer of rate 50%

Add a dense layer with activation function 'softmax'.

#Configuring the model

Compile model ( loss = categorical_crossentropy, adam optimizer, metrics =
'accuracy')

#Training the model

Set epochs to desired number

Model.fit(input = x_train, y_train, batch_size, epochs, validate with x_test
and y_test)
#Save model
Model.save()
```

## **7.3 Pseudo Code for Web App**

```
app.py
#Importing required libraries
    Import>Flask, keras.models, numpy, PIL, puttsx3
#Initialize Flask
    App= Flask(_name_)
#define no of classes in a dictionary
    Classes = {Class label: 'Name of the class' }
#define function to predict
    Load model
    Open image
    Resize image
    Append image data onto the array
    Model.predict(img)
    Return predicted Class
#define various pages/navigation
    Login()
    First page()
#define upload function to predict picture
    Get image via POST request method
    call function to predict
    #Use pyttsx to send alert
    Set engine.runandWait()
Return Result()
```

## CHAPTER-8

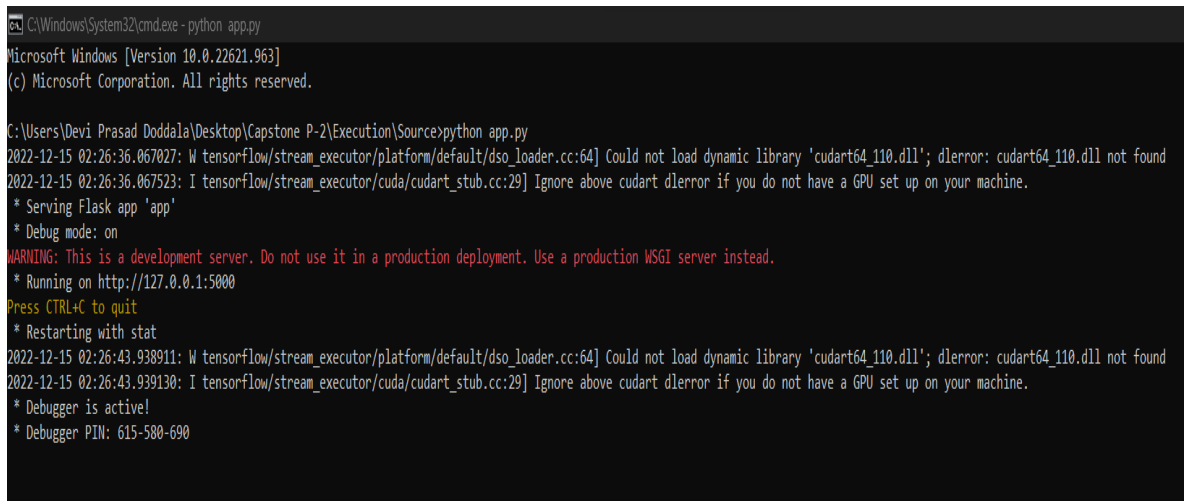
# RESULTS AND DISCUSSION

### 8.1 Testing

The model is tested with the existing data available from the GTSRB dataset. The model shows an accuracy of 95% when tested with the GTSRB dataset.

The web app shows no signs of malfunctioning even when used continuously. The web app navigates to pages normally and quickly with no delay. The web app logs in correctly when entered the correct credentials. The application loads all the graphs correctly and shows the prediction analysis report.

### 8.2 Screenshots of the web app

A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\System32\cmd.exe - python app.py". The window content shows the following text:

```
Microsoft Windows [Version 10.0.22621.963]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Devi Prasad Doddala\Desktop\Capstone P-2\Execution\Source>python app.py
2022-12-15 02:26:36.067027: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll not found
2022-12-15 02:26:36.067523: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
2022-12-15 02:26:43.938911: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll not found
2022-12-15 02:26:43.939130: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
* Debugger is active!
* Debugger PIN: 615-580-690
```

*Figure 16. Command prompt running the app.py*

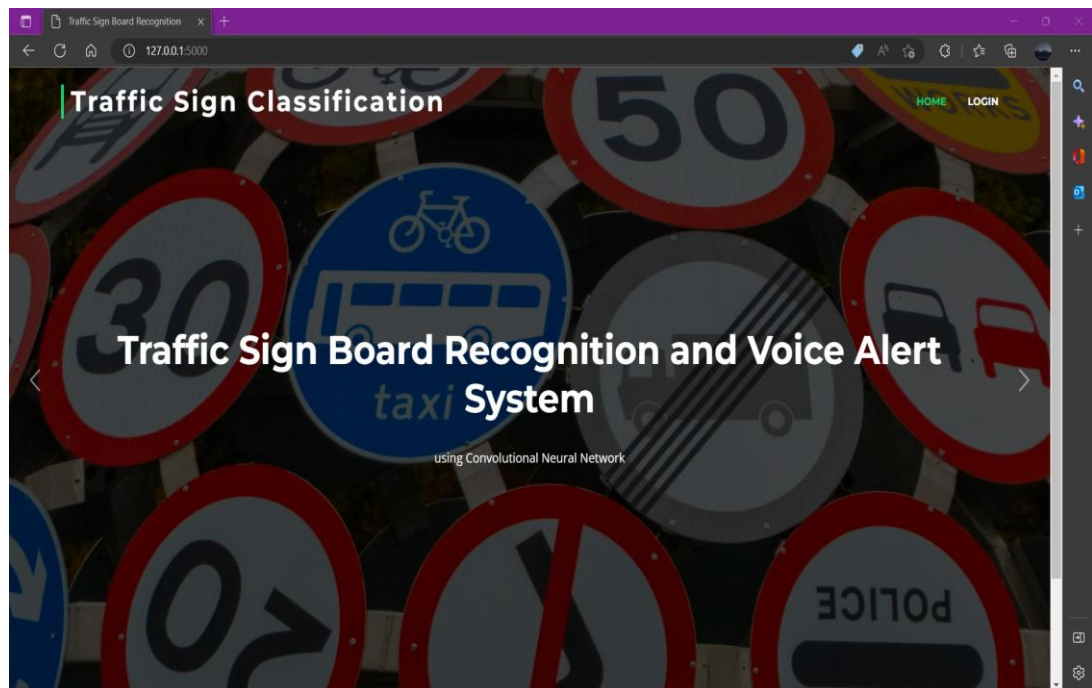


Figure 17. First Opening Page of Web App

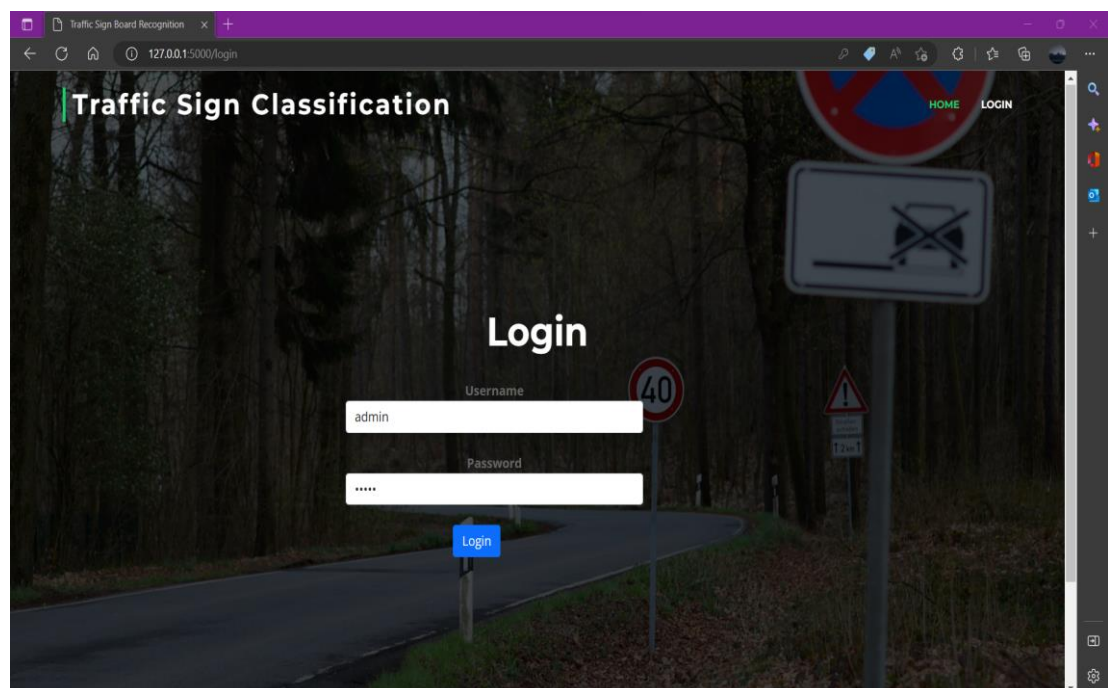


Figure 18. Login Page of Web App



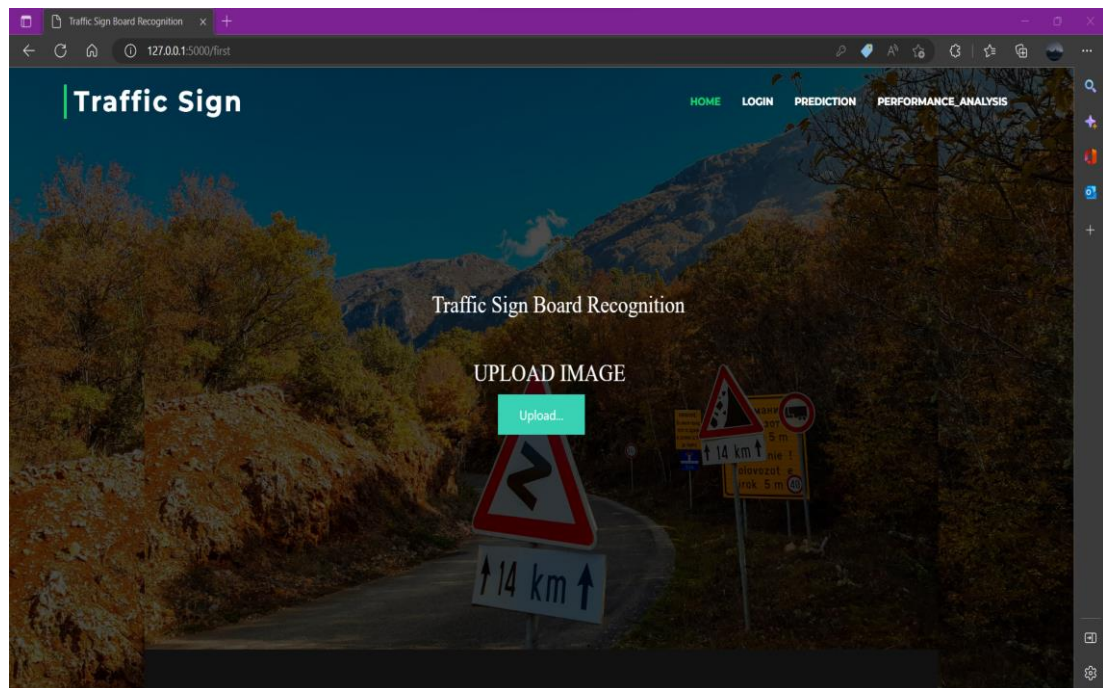


Figure 19. Upload Page/Prediction Page

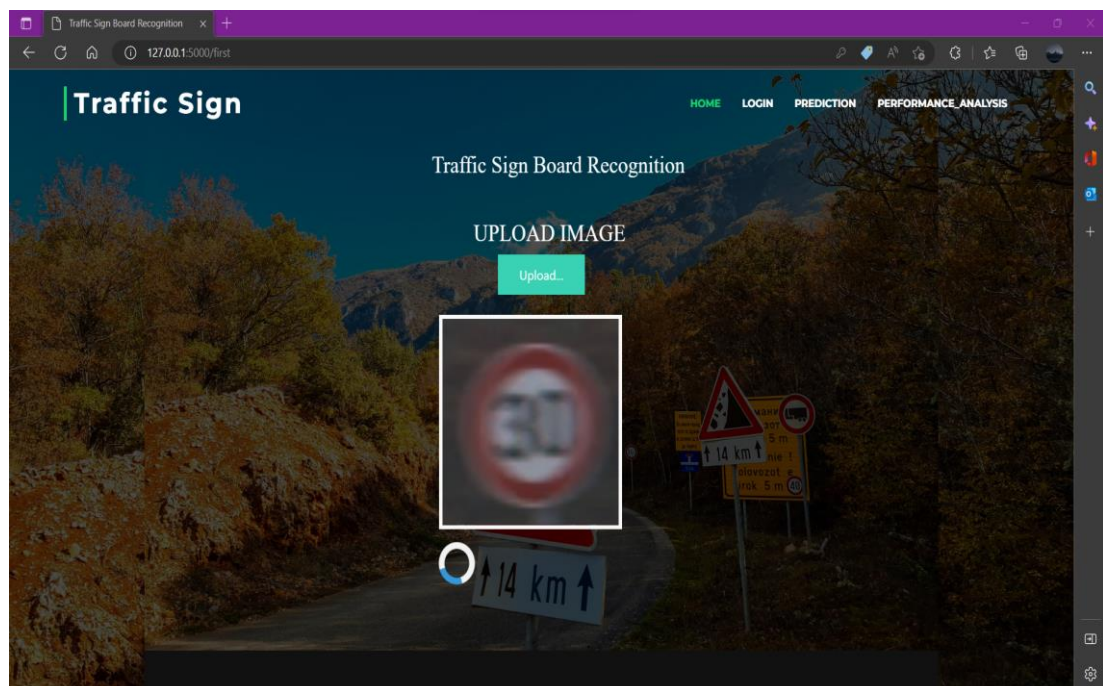


Figure 20. Prediction in Progress



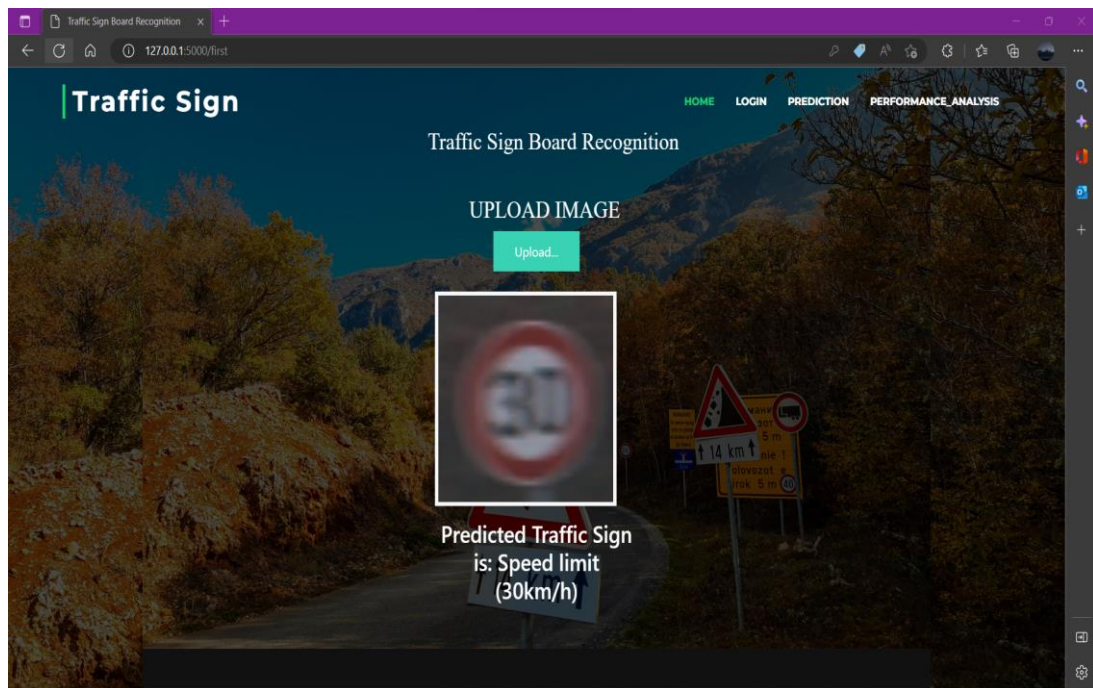


Figure 21. Sign Predicted

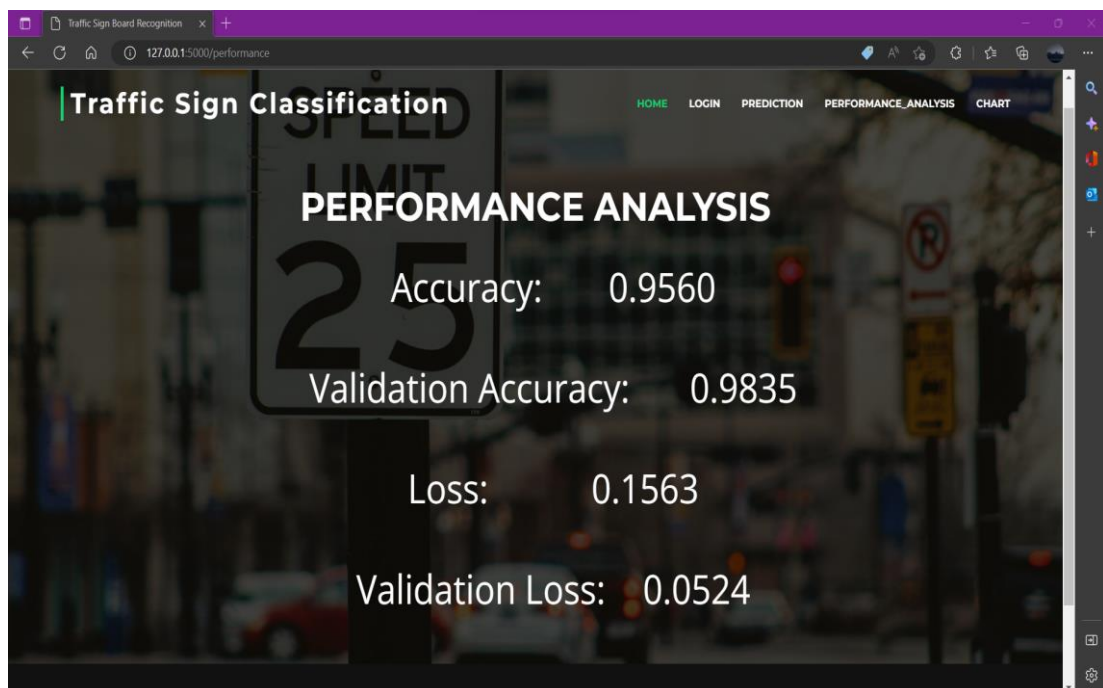


Figure 22. Performance Analysis

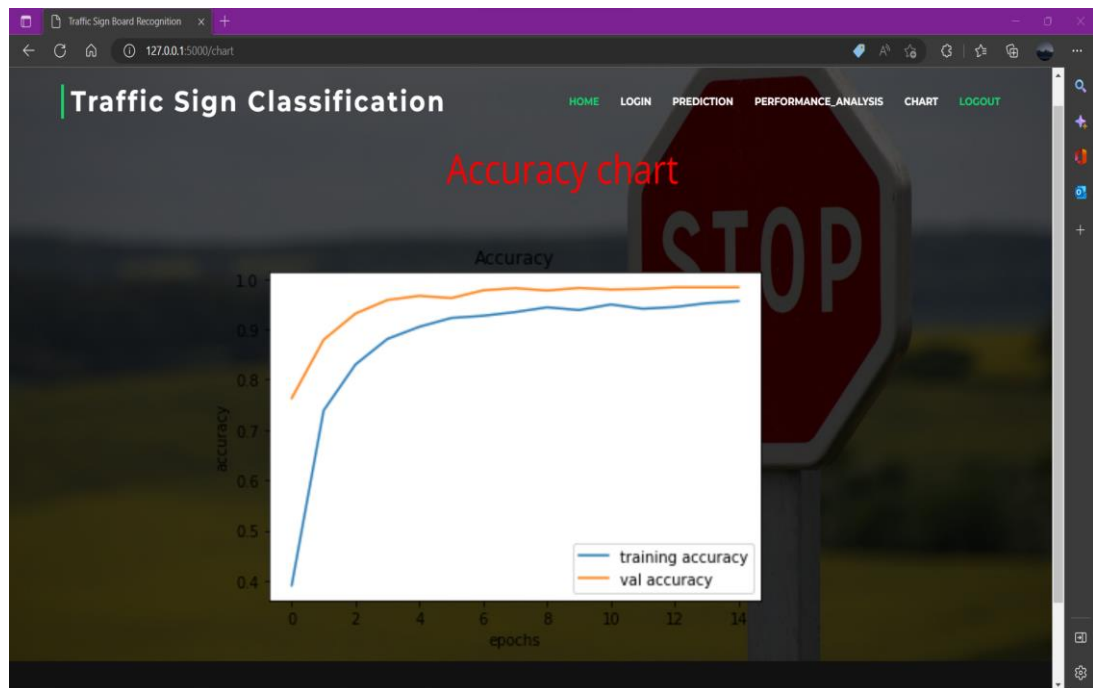


Figure 23. Accuracy Chart

### 8.3 Accuracy and Loss Plots

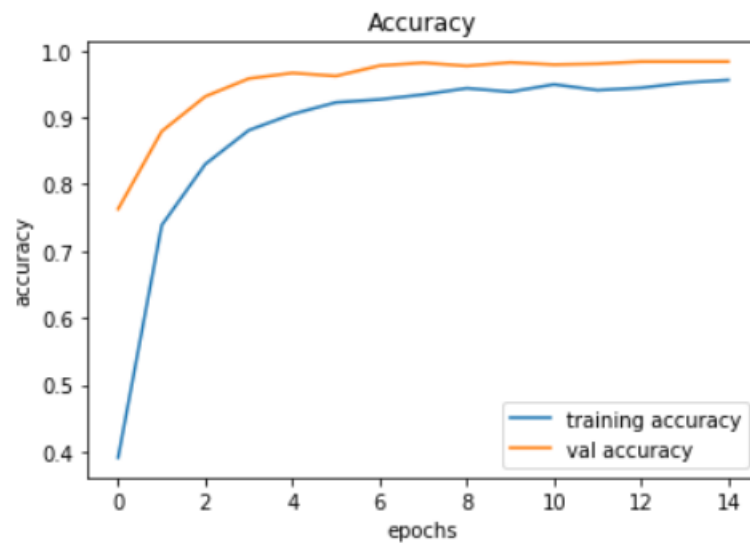


Figure 24. Accuracy Plot

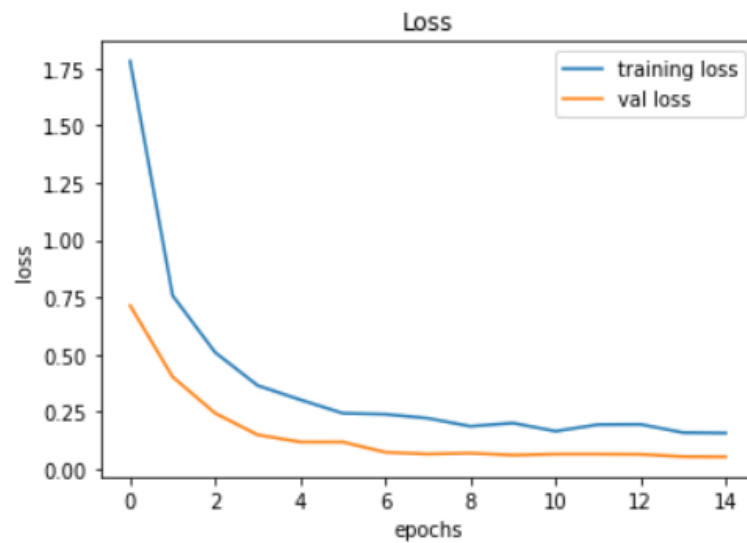


Figure 25. Loss Plot

## 8.4 Jupyter Notebook Testing

```
plot, prediction = test_on_img(r'C:\Users\Devi Prasad Doddala\Desktop\Capstone P-2\Execution\Source\Test\00022_00003_00006.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

1/1 [=====] - 0s 17ms/step  
Predicted traffic sign is: Bumpy Road

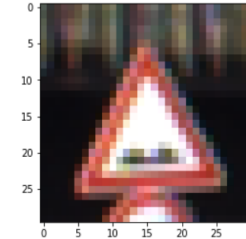


Figure 26. Jupyter Notebook test screenshot

## **8.5 Results and Discussion**

The neural network that has been trained with a total of 12 layers of which 4 are convolution layers ,2 max pooling layers with the combination of flatten layer, dense layers and dropout layers has proved to give a good result in relation with other layers such Resnet34. We should also keep in mind that this was done with the same the dataset only. It obtains an accuracy of about 95%.

The trained neural network can perform well beyond its potential if there is a proper dataset and if the data augmentation techniques used are on point and also tweaking the model layers will also affect the accuracy of the model.

## CHAPTER-9

### CONCLUSION AND FUTURE WORK

#### 9.1 Conclusion

The Convolutional Neural Network is used to construct the Traffic Sign Board Detection and Voice Alert System. Various CNN models were investigated, and the one with the greatest accuracy on the GTSRB dataset was implemented. The introduction of several classifications for each traffic sign has aided in the model's accuracy. Following the detection of the sign, an audio message is broadcast to warn the driver, assisting him or her in making suitable judgments. This study is an important achievement in the realm of driving since it would make the driver's job easier without sacrificing safety. This system may also be simply deployed without requiring a lot of hardware, expanding its reach.

#### 9.2 Future work

The prototype may be developed to incorporate an integrated alert system with a camera in the centre of the vehicle. Additionally, the functionality of obtaining an estimated time for reaching that specific traffic sign may be included. This system may also be enhanced to identify traffic lights and therefore notify the user of the time required to reach that signal as well as its condition. The user may thus schedule their journey start time properly and cross all signals without having to wait. Driver verification will also be performed with the assistance of an API that provides information on the licence holder and the licence number.

## REFERENCES/BIBLIOGRAPHY

- [1] Tabernik, D. and Skočaj, D., 2020. Deep learning for large-scale traffic-sign detection and recognition. *IEEE transactions on intelligent transportation systems*, 21(4), pp.1427-1440.
- [2] V. Balali, A. A. Rad, and M. Golparvar-Fard, "Detection, classification, and mapping of U.S. traffic signs using Google street view images for roadway inventory management," *Vis. Eng.*, vol. 3, no. 1., 2015, p. 15.
- [3] Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic sign detection and classification in the wild," in *Proc. CVPR*, Jun. 2016, pp. 2110–2118.
- [4] A. Ellahyani, M. El Ansari, I. El Jaafari, and S. Charfi, "Traffic sign detection and recognition using features combination and random forests," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 1, 2016, pp. 683–693.
- [5] Huang, Z., Yu, Y., Gu, J. and Liu, H., 2016. An efficient method for traffic sign recognition based on extreme learning machine. *IEEE transactions on cybernetics*, 47(4), pp.920-933.
- [6] Li, H., Sun, F., Liu, L. and Wang, L., 2015. A novel traffic sign detection method via color segmentation and robust shape matching. *Neurocomputing*, 169, pp.77-88.
- [7] Haloi, M., 2015. Traffic sign classification using deep inception based convolutional networks. *arXiv preprint arXiv:1511.02992*.
- [8] Stallkamp, J., Schlipsing, M., Salmen, J. and Igel, C., 2011, July. The German traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, IEEE. pp. 1453- 1460.
- [9] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [10] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778.

## **APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS**

1. CNN – Convolutional Neural Networks
2. SVM – Support Vector Machines
3. TSDR – Traffic Sign Detection and Recognition
4. HSV – Hue, Saturation, and Value
5. LBP – Local Binary Pattern
6. Hog – Histogram of Oriented Gradients
7. LSS – Local Self-Similarity
8. ELM – Extreme Learning Machine
9. GTSRB – German Traffic Sign Board Recognition Benchmark
10. TSR – Traffic Sign Recognition
11. ADAS – Advanced Driver Assistance Systems

## CS\_report\_plagarism-1.docx

### ORIGINALITY REPORT

<b>6%</b>	<b>4%</b>	<b>2%</b>	<b>2%</b>
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	<b>www.analyticsvidhya.com</b> Internet Source	<b>2%</b>
<b>2</b>	<b>www.geeksforgeeks.org</b> Internet Source	<b>1%</b>
<b>3</b>	<b>www.irjmets.com</b> Internet Source	<b>1%</b>
<b>4</b>	<b>Submitted to University of Salford</b> Student Paper	<b>1%</b>
<b>5</b>	<b>abhishekghunjariya-ds-webscraping.blogspot.com</b> Internet Source	<b>1%</b>
<b>6</b>	<b>Ayoub ELLAHYANI, Mohamed EL, Ilyas EL, Said CHARFI. "Traffic Sign Detection and Recognition using Features Combination and Random Forests", International Journal of Advanced Computer Science and Applications, 2016</b> Publication	<b>&lt;1%</b>
<b>7</b>	<b>Submitted to KEMUDA Institute</b> Student Paper	<b>&lt;1%</b>