NEW HORIZON
COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

A

MINI PROJECT REPORT

ON

"TRAFFIC SIGNS RECOGNITION USING CNN"

Submitted in the partial fulfillment of the requirements in the 6th semester of

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

BY

SNEHA B K

1NH17IS103

Under the guidance of

Mrs. MOUNICA B
Sr. Assistant Professor,
Dept. of ISE, NHCE

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

NEW HORIZON COLLEGE OF ENGINEERING

(Autonomous College Permanently Affiliated to VTU, Approved by AICTE, Accredited by NBA & NAAC with 'A' Grade)
New Horizon Knowledge Park, Ring Road, Bellandur Post, Near Marathalli,
Bangalore-560103, INDIA

NEW HORIZON
COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

I hereby certify that, the report entitled **"Traffic Signs Recognition using CNN"** as a part of Mini Project Component in partial fulfillment of the requirements during 6th semester Bachelor of Engineering in Information Science and Engineering during the year 2019-2020 is an authentic record of my own work carried out by Sneha B K (1NH17IS103), a bonafied student of NEW HORIZON COLLEGE OF ENGINEERING.

Name & Signature of Student

(Ms. Sneha B K)

Name & Signature of Guide

(Mrs. Mounica B)

Name & signature of HOD

(Dr. R J Anandhi)

ACKNOWLEDGEMENT

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped me in carrying out this project. I would like to take an opportunity to thank them all.

First and foremost I thank the management, **Dr. Mohan Manghnani,** Chairman, New Horizon Educational Institutions for providing necessary infrastructure and creating good environment.

I would like to thank **Dr. Manjunatha**, Principal, New Horizon College of Engineering, Bengaluru, for his constant encouragement and facilities extended to us towards completing my project work.

I extend my sincere gratitude to **Dr. R J Anandhi**, Head of the Department, Information Science and Engineering, New Horizon College of Engineering, Bengaluru for her valuable suggestions and expert advice.

I deeply express my sincere gratitude to my guide **Mrs. Mounica B, Sr. Assistant Professor**, Department of ISE, NHCE, Bengaluru, for his/her able guidance, regular source of encouragement and assistance throughout this project.

I thank my Parents, and all Faculty members of Department of Information Science and Engineering for their constant support and encouragement.

Last, but not the least, I would like to thank my peers and friends who provided me with valuable suggestions to improve my project.

Sneha B K

(1NH17IS103)

ABSTRACT

The traffic sign recognition is an integral part of Advanced Driver Assistance System (ADAS). Traffic signs give data about the traffic rules, street conditions and course bearings and help the drivers for better and safe driving. We heard about the self-driving vehicles in which the traveler can completely depend on the vehicle for traveling.

CNN (Convolutional Neural Network) is used to classify images into their respective categories, by building a CNN model.

Building the traffic sign classification model includes following steps:

- Explore the dataset
- Build a CNN model
- Train and validate the model
- Test the model with test dataset

The proposed approach consists of two subsystems for detection and recognition. Traffic sign recognition can help the driver to make a right decision at the right time for safe driving. In this project, deep neural network model is built that can classify traffic signs present in the image into different categories. The project utilizes datasets which are first trained, generate CNN model, tests and classifies. A graphical user interface for our traffic signs classifier is built with Tkinter. The language that will be used is Python. It is feasible and efficient. Proposed system which will classify and recognize the traffic signs.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

In a recent day's there is a great deal of consideration being given to the capacity of the vehicle to drive itself. Important aspects for a self-driving vehicle is the ability for it to detect traffic signs in order to provide safety and security for the people not only inside the vehicle but also outside of it.

There are few unique kinds of traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc. Traffic signs order is the way toward recognizing which class a traffic sign has a place with. In this Python project, we will build a deep neural network model that can characterize traffic signs present in the picture into various classes. With this model, we can peruse and comprehend traffic signs which are a significant undertaking for every self-sufficient vehicle. This undertaking utilizes CNN and Keras library with giving 95% precision in Traffic Sign Recognition. Along these lines, for accomplishing precision in this technology, the vehicles should be able to interpret traffic signs and make needs accordingly.

In this "Traffic Sign recognition" project the fundamental objective is to design and develop a computer-based system which can automatically detect the road signs to give assistance to the client or the machine so that they can take suitable actions.

1.1 Purpose of Study

The reason for 'traffic sign recognition' project is to classify the detected traffic signs to their specific sub-classes. Traffic sign images classification and recognition is for High quality recordings. In any case, the picture handling is tedious and requires some computerization to spare the ideal opportunity for picture identification and checking. A framework has been created to recognize and check dynamic traffic signs precisely and productively.

It may be chosen because it is thought to provide following advantages:

1.This programming is time effective.

2. Detection of various images of traffic signs.

3. Classification of distinguished traffic signs.

4. Recognition of traffic signs.

1.2 Problem Statement

“To design and implement Traffic sign Recognition using CNN classifier and Keras”

1.3 Motivation of project

"Traffic Sign Recognition" is an undertaking which is created to give better help to classify and recognize the road signs or traffic signs. This application venture gives proper accuracy of traffic or road sign image. In this product OpenCV-python library is utilized to break down and control the picture. Picture handling is to separate significant information from pictures. Primary concern of picture preparing is to adjust pictures or recordings into wanted way. At the end of the day, picture handling is called as changing and examining pictorial data of pictures or recordings.

We can improve the efficiency of the system thus overcome the drawbacks of the existing system.

- Less human error
- High security
- Data consistency

Achieving this objective is difficult using manual system as the information is scattered, can be collecting relevant information may be very time consuming. All these problems are solved by using this project.

Chapter 2

System Requirement and Language used

2.1 Hardware and software requirements

Hardware requirements

Processor - Intel Core i5

Speed - 1.8 GHz

RAM - 256 MB (min)

Hard Disk - 10 GB

Software requirements

Operating System - windows 7/8.1/10

Programming Language - Python

Compiler - Python Idle

2.2 About the Language

The project “Traffic Sign Recognition” is implemented using Python Programming Language. With python language OpenCV, Pandas, numpy, keras, tenserflow, Matplotlib, Scikit-learn, PIL and image classification. libraries are used to implement this project. Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.

Chapter 3

System Design

3.1 Architecture

In this "Traffic Sign Recognition" project utilizes datasets which are first trained, generate CNN model, tests and classifies. The figure3.1, gives an overview of the steps that explore datasets, Build the CNN model, train, validate and test the model with test dataset.

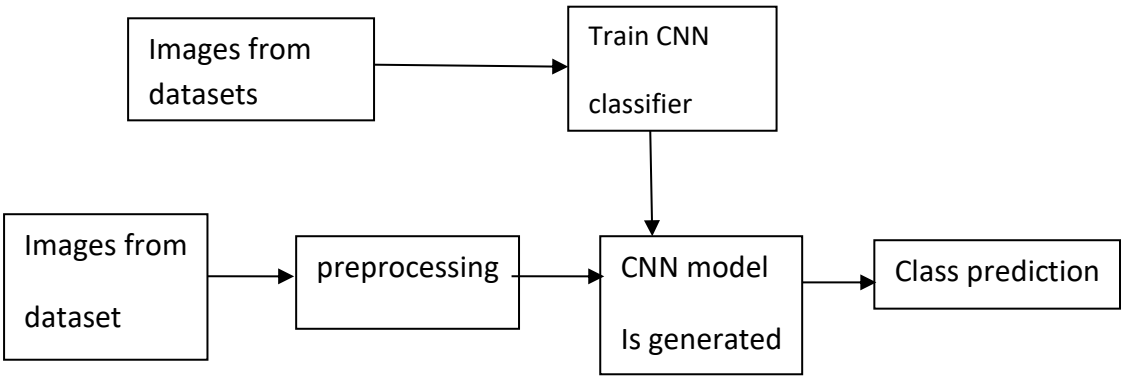

Fig: 3.1 Process of Traffic sign recognition using datasets.

"Traffic sign recognition" project uses dataset which has two tasks first includes Building Traffic sign classification model, second includes Building Graphical user interface for traffic sign classifier.

Traffic sign classification model building done by using four steps:

1. Exploring the dataset
2. Building a CNN model
3. Training and Validating the model
4. Testing the model with test dataset

Step 1: Exploring the dataset

In provided dataset, train folder contains 43 folders each representing a different class. OS module iterate all the classes and append images and their respective labels in the data and labels list. Here PIL library is used to open image content into an array. Then that list converted into Numpy arrays for feeding to the model. To split training and testing data we use the train_test_split() method from sklearn package and to convert the labels present in y_train and t_test into one-hot encoding we use to_categorical method from the keras.utils package.

Step 2: Building CNN model

CNN model is build to classify the images into their respective categories.

The architecture of CNN model is:

- 2 Conv2D layer (filter=32, kernel_size=(5,5), activation="relu")
- MaxPool2D layer (pool_size=(2,2))
- Dropout layer (rate=0.25)
- 2 Conv2D layer (filter=64, kernel_size=(3,3), activation="relu")
- MaxPool2D layer (pool_size=(2,2))
- Dropout layer (rate=0.25)
- Flatten layer to squeeze the layers into 1 dimension
- Dense Fully connected layer (256 nodes, activation="relu")
- Dropout layer (rate=0.5)
- Dense layer (43 nodes, activation="softmax")

Step 3: Training and validating the model

After building the CNN model architecture, by using model.fit() method in this step we can train the model. After training the model by using Matplotlib library graph for accuracy and the loss will be plotted.

Step 4: Testing the model with test dataset

The provided dataset contains a test folder and in a test.csv file, we have the details identified with the picture way and their particular class labels. We separate the picture path and names utilizing pandas. At that point to predict the model, we need to resize our pictures to 30×30 pixels and make a numpy exhibit containing all picture information. From the sklearn.metrics, we imported the accuracy_score and observed how our model anticipated the actual labels. At the end, trained model is saved using the Keras model.save() function.

Now second task, Building Graphical user interface for traffic sign classifier. Here we are using Tkinter which is a GUI toolkit in the standard python library.

In this first step is to load the trained model 'traffic_classifier.h5' using Keras. Then construct the GUI for uploading the picture and a button is used to characterize which calls the classify() function. The classify() function is changing over the picture into the element of shape (1, 30, 30, 3). This is because to predict the traffic sign we need to give the similar measurement we have utilized when constructing the model. Then predict the class, the model.predict_classes(image) returns us a number between (0-42) which represents the class it has a place with. Dictionary is used to label all traffic signs and to get the information about the class.

3.2 Flowchart

A Flowchart is a type of diagram that represents an algorithm, process. It is diagrammatic representation of an algorithm.

3.2.1 Flow diagram for detection of vehicle

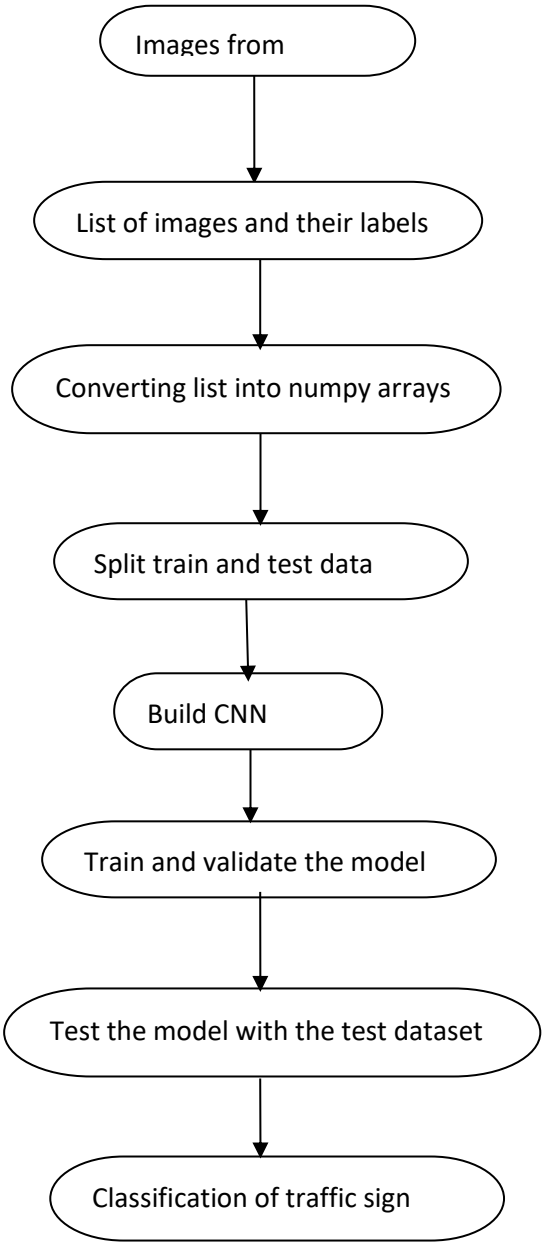

Fig 3.1: Flow diagram for Recognition of traffic sign

3.3 Code and Implementation

3.3.1 Implementation of building traffic sign classification model

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import cv2

import tensorflow as tf

from PIL import Image

import os

from sklearn.model_selection import train_test_split

from keras.utils import to_categorical

from keras.models import Sequential, load_model

from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

data = []

labels = []

classes = 43

cur_path = os.getcwd()

for i in range(classes):

    path = os.path.join(cur_path,'train',str(i))

    images = os.listdir(path)
```

```
    for a in images:
        try:
            image = Image.open(path + '\\'+ a)
            image = image.resize((30,30))
            image = np.array(image)
            #sim = Image.fromarray(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error loading image")
data = np.array(data)
labels = np.array(labels)
print(data.shape, labels.shape)
#Splitting training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
```

```
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))

model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Dropout(rate=0.25))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))

model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Dropout(rate=0.25))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

model.add(Dropout(rate=0.5))

model.add(Dense(43, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

epochs = 15

history = model.fit(X_train, y_train, batch_size=32, epochs=epochs,
validation_data=(X_test, y_test))

model.save("my_model.h5")

plt.figure(0)

plt.plot(history.history['accuracy'], label='training accuracy')

plt.plot(history.history['val_accuracy'], label='val accuracy')

plt.title('Accuracy')

plt.xlabel('epochs')

plt.ylabel('accuracy')
```

```
plt.legend()

plt.show()

plt.figure(1)

plt.plot(history.history['loss'], label='training loss')

plt.plot(history.history['val_loss'], label='val loss')

plt.title('Loss')

plt.xlabel('epochs')

plt.ylabel('loss')

plt.legend()

plt.show()

from sklearn.metrics import accuracy_score

y_test = pd.read_csv('Test.csv')

labels = y_test["ClassId"].values

imgs = y_test["Path"].values

data=[]

for img in imgs:

    image = Image.open(img)

    image = image.resize((30,30))

    data.append(np.array(image))

X_test=np.array(data)

pred = model.predict_classes(X_test)
```

```
from sklearn.metrics import accuracy_score

print(accuracy_score(labels, pred))
```

3.3.2. Implementation of building graphical user interface for traffic sign classifier

```
import tkinter as tk

from tkinter import filedialog

from tkinter import *

from PIL import ImageTk, Image

import numpy

from keras.models import load_model

model = load_model('traffic_classifier.h5')

classes = { 1:'Speed limit (20km/h)',

        2:'Speed limit (30km/h)',

        3:'Speed limit (50km/h)',

        4:'Speed limit (60km/h)',

        5:'Speed limit (70km/h)',

        6:'Speed limit (80km/h)',

        7:'End of speed limit (80km/h)',

        8:'Speed limit (100km/h)',

        9:'Speed limit (120km/h)',
```

10:'No passing',

11:'No passing veh over 3.5 tons',

12:'Right-of-way at intersection',

13:'Priority road',

14:'Yield',

15:'Stop',

16:'No vehicles',

17:'Veh > 3.5 tons prohibited',

18:'No entry',

19:'General caution',

20:'Dangerous curve left',

21:'Dangerous curve right',

22:'Double curve',

23:'Bumpy road',

24:'Slippery road',

25:'Road narrows on the right',

26:'Road work',

27:'Traffic signals',

28:'Pedestrians',

29:'Children crossing',

30:'Bicycles crossing',

```
        31:'Beware of ice/snow',
        32:'Wild animals crossing',
        33:'End speed + passing limits',
        34:'Turn right ahead',
        35:'Turn left ahead',
        36:'Ahead only',
        37:'Go straight or right',
        38:'Go straight or left',
        39:'Keep right',
        40:'Keep left',
        41:'Roundabout mandatory',
        42:'End of no passing',
        43:'End no passing veh > 3.5 tons' }
top=tk.Tk()
top.geometry('800x600')
top.title('Traffic sign classification')
top.configure(background='#CDCDCD')
label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = Label(top)
def classify(file_path):
   global label_packed
```

```
    image = Image.open(file_path)
    image = image.resize((30,30))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    print(image.shape)
    pred = model.predict_classes([image])[0]
    sign = classes[pred+1]
    print(sign)
    label.configure(foreground='#011638', text=sign)
  def show_classify_button(file_path):
classify_b=Button(top,text="ClassifyImage",command=lambda:classify(file_path),padx=10,pady=5)
classify_b.configure(background='#364156',foreground='white',font=('arial',10,'bold'))
classify_b.place(relx=0.79,rely=0.46)
def upload_image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)
        uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)))
        im=ImageTk.PhotoImage(uploaded)
        sign_image.configure(image=im)
        sign_image.image=im
```

```
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass
upload=Button(top,text="Upload an image",command=upload_image,padx=10,pady=5)
upload.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True)
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="Know Your Traffic Sign",pady=20, font=('arial',20,'bold'))
heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

Chapter 4

RESULTS AND DISCUSSION

4.1 Summary of result obtained

This project has been developed to classify and recognize the traffic or road signs with more accuracy and efficiently. After training the model, graph plotted for accuracy reading. By using dictionary traffic signs are labeled and images are classified and recognized by GUI interface.

4.2 Output (Snapshots)

Fig 4.1: output snapshot 1: Training datasets

Fig 4.2: output snapshot 2: Training datasets

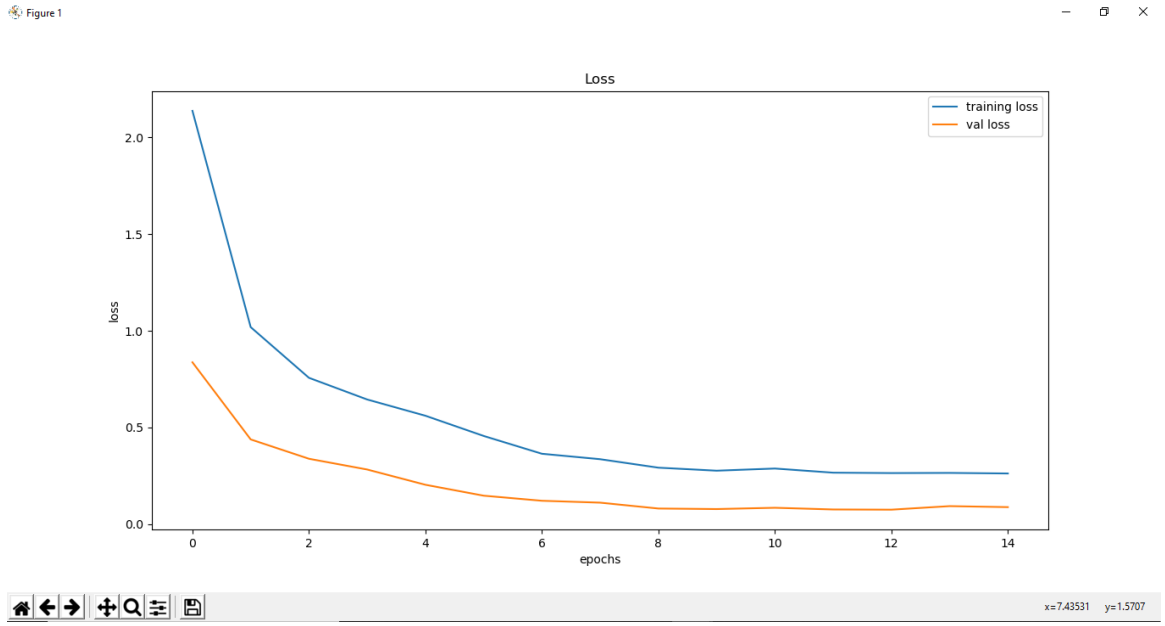

Fig 4.3: output snapshot 3: Graph of Accuracy

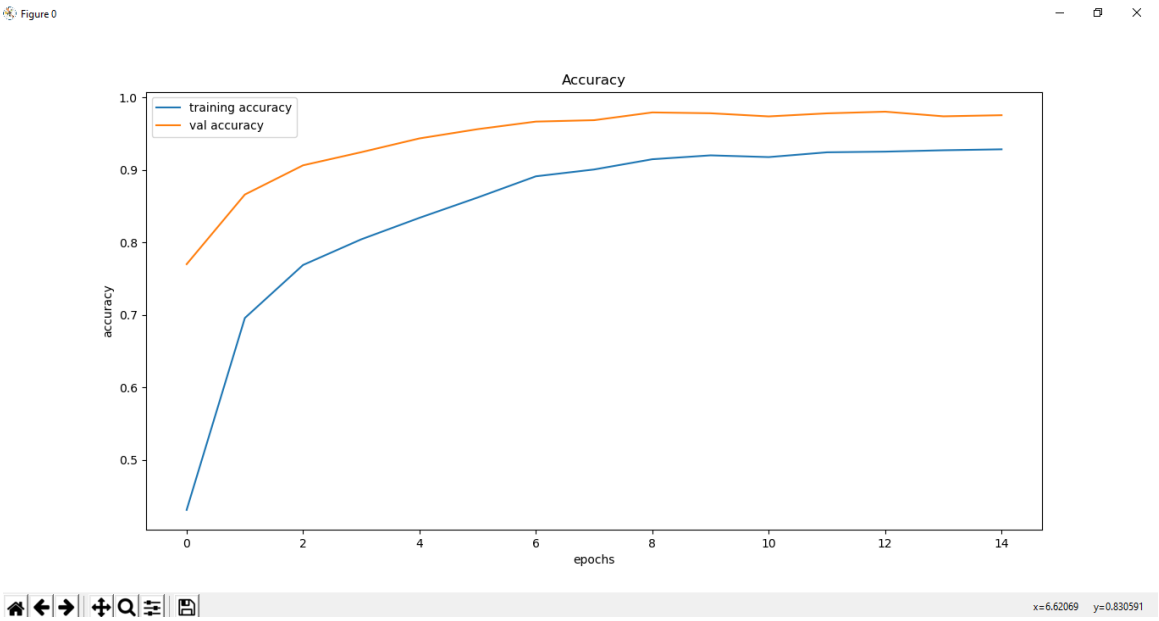

Fig 4.4: output snapshot 4: Graph of Loss

Fig 4.5: output snapshot 5: GUI to upload image

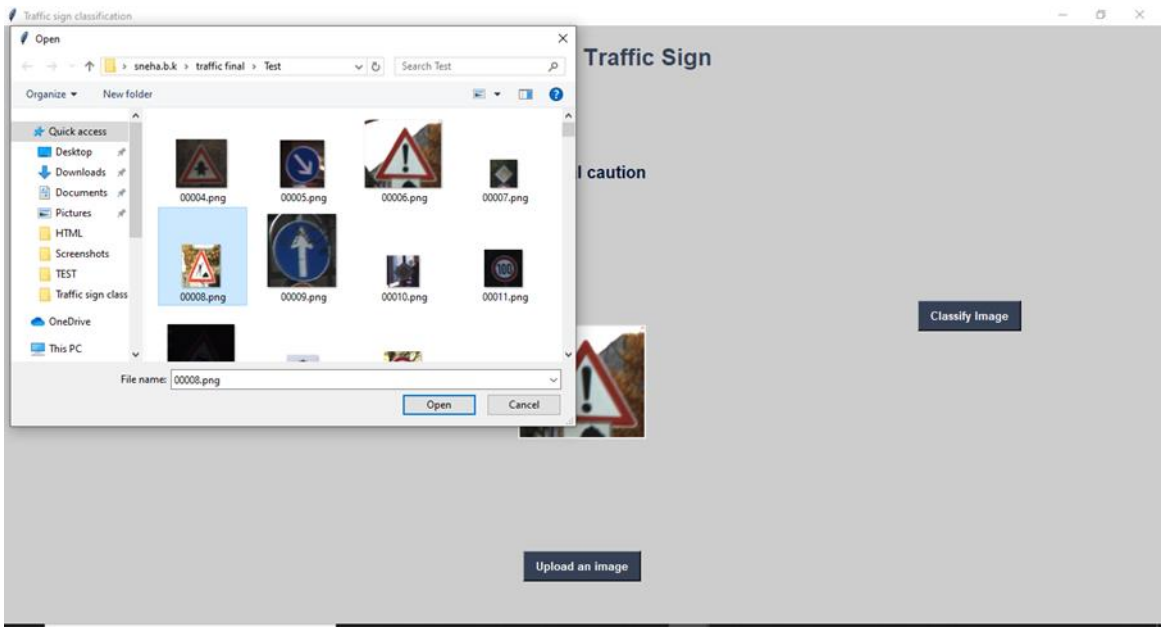

Fig 4.6: output snapshot 6: GUI for upload image from test folder

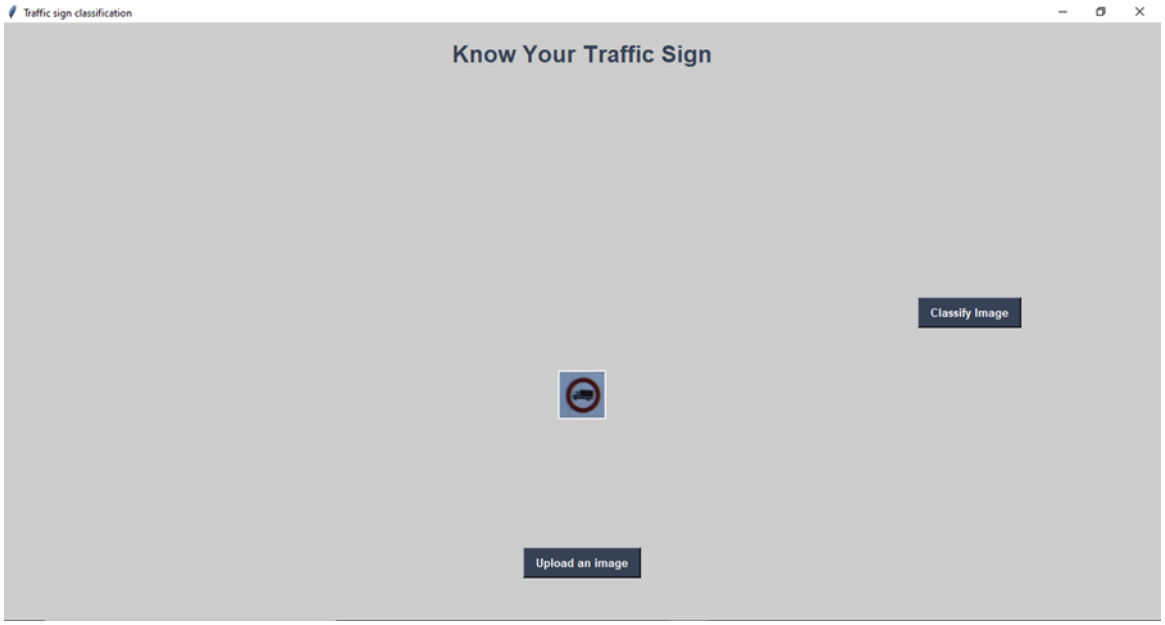

Fig 4.7: output snapshot 7: GUI for classify image

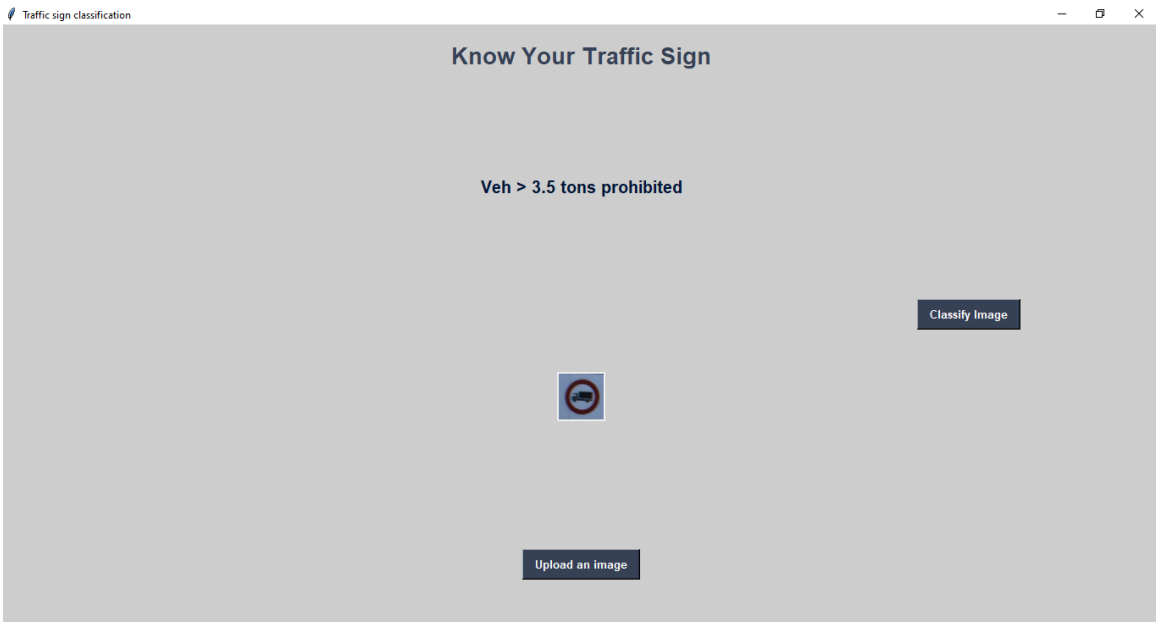

Fig 4.8: output snapshot 8: Classification of image

Fig 4.9: output snapshot 9: Classification of image

Chapter 5

CONCLUSION

This application-project reduces the manual work, maintaining accuracy, increasing efficiency, reduces the search area and saving time. We are developing a proposed system which will recognize traffic or road signs. Different signs classified using CNN classifier. This project was developed using OpenCV and Matlab, numpy, keras, pandas, tenserflow, Tkinter and PIL libraries which made the project comparatively easy to implement and understand. Here Traffic Sign Recognition is done by Convolutional Neural Network, which is best for image classification purposes. Therefore, the above application proves to be efficient.

References

[1]. "Cracking the Coding Interview: 189 Programming Questions and Solutions" by Gayle Laakmann McDowell.

[2]. https://data-flair.training/blogs/python-project-traffic-signs-recognition/

[3]. https://www.ijeecse.com/ICSCAAIT_083.pdf

[4]. www.geeksforgeeks.org

[5]. S. Estable, J. Schick, F. Stein, R. Janssen, R. Ott, W. Ritter, and Y.-J. Zheng, "A real-time traffic sign recognition system," in Proc. IEEE Intelligent Vehicles '94 Symposium, 1994, pp. 213–218