**WARNING SYSTEM FOR DRIVERS**

15CSE387 OPEN LAB

PROJECT REPORT

*Submitted by*

M.SARATH CHANDRA BL.EN.U4CSE17072

P.NITHIN BL.EN.U4CSE17097

M.LAKSHMAN KRUTHIK BL.EN.U4CSE17077



AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

BANGALORE 560 035

06-2020

**Annexure 1**

**Abstract :**

Leading a good and safe life is the primary objective of any person. In the fastly growing and developing society everyone is in a rush and are behind getting their jobs done. In order to achieve their tasks everyone is using a vehicle these days which has resulted in a society where around 25 million vehicles are being produced per year. This massive production and also the use of vehicles is not only degrading the environment by increasing the pollution but also this is resulting in increasing the risk and probability of fatality and destruction due to vehicle accidents. In these accidents most of the cases are mainly due to the reckless behaviour of the vehicle drivers which can have many reasons like restlessness for the driver etc. To overcome this problem we came up with a solution to build a software that can assist a driver by producing messages and creating awareness about the surroundings. This can reduce the probability of accidents due to vehicles which can save many lives and help in achieving the primary objective of any person i.e, to live a safe life. To develop this software we use the concepts of computer vision and Deep Learning techniques which empowers our software by giving it the ability to distinguish the objects in the surroundings. This software can produce audio messages that have the information about the traffic signs and the crowd intensity level present in the area where he is approaching. The range of the camera upto which it can analyse the situation is about ten meters. This software can be developed and can be used in an embedded system that can be mounted on a vehicle which can be used in real life.

TABLE OF CONTENTS

CHAPTER 1 [INTRODUCTION](#_heading=h.2et92p0)

[1.1 OBJECTIVE……………………………………………… ……………………..](#_heading=h.tyjcwt)

[1.2 MOTIVATION………………………………………………… ……………….](#_heading=h.3dy6vkm)

1.3 PROBLEM STATEMENT………………………………………………………10

1.4 PROPOSED SOLUTION……………………………………………………...11

[CHAPTER 2 USE CASE DIAGRAM AND DESCRIPTION……………………](#_heading=h.1t3h5sf)

[CHAPTER 3 IMPLEMENTATION …………………………………… ……….](#_heading=h.4d34og8)

[CHAPTER 4](#_heading=h.2s8eyo1) [RESULTS 32](#_heading=h.17dp8vu)

[CHAPTER 5](#_heading=h.3rdcrjn)  [CONCLUSION AND FUTURE SCOPE 34](#_heading=h.26in1rg)

[CHAPTER 6](#_heading=h.lnxbz9) [REFERENCES 36](#_heading=h.35nkun2)

**INTRODUCTION**

In the fastly growing and developing society everyone is in a rush and are behind getting their jobs done. In order to achieve their tasks everyone is using a vehicle these days which has resulted in a society where around 25 million vehicles are being produced per year. This massive production and also the use of vehicles is not only degrading the environment by increasing the pollution but also this is resulting in increasing the risk and probability of fatality and destruction due to vehicle accidents. In these accidents most of the cases are mainly due to the reckless behaviour of the vehicle drivers which can have many reasons like restlessness for the driver etc. Road accidents have become very common nowadays. As more and more people are buying automobiles, the incidences of road accidents are just increasing day by day. Furthermore, people have also become more careless now. Not many people follow the traffic rules. Especially in big cities, there are various modes of transports. Moreover, the roads are becoming narrower and the cities have become more populated.

Thus, road accidents are bound to happen. You pick up a newspaper and you will find at least one or two news about road accidents daily. They cause loss of life as well as material. People need to be more careful when on the road, no matter which mode of transport you are from. Even the ones on foot are not safe because of the rise in these incidences. Everyday people witness accidents in the news, from relatives and even with their own eyes. This results in economical loss, materialistic loss and finally fatality which is something that can't be brought back. Development is certainly needed and that is how the society should run but this development and safety of the people should go hand in hand, otherwise there would only be machines left in the world.

We need to prevent road accidents to decrease the death rate. Every year thousands of people lose their lives to road accidents. Children must be taught from an early age about traffic rules. They must be taught the value of life and how they can safeguard it. Moreover, the government must pass more stringent laws for people who disobey traffic rules. They must find people heavily or take strict action when found guilty of breaking these laws irrespective of gender.Similarly, parents must set an example for the younger ones by not using phones while driving. Also, they must always wear their helmets and seatbelts to avoid the chances of an accident.

**Objective**

To prevent all this fatality and disasters that happen in the society the drivers should get some kind of assistance that guide them or suggest them whilst driving. It's never a feasible or efficient idea to expect a person to be with you when you are driving to assist you. Hence a system can be made that can provide assistance to the driver. The system should be able to analyse the situation in the surrounding areas of the vehicle and should be able to decide whether there is a crowd in that area. The system should be able to take the input from the attached input camera device and should process the input to produce decisions that it makes by analysing the input in runtime.

**Motivation:**

In the fastly growing and developing society everyone is in a rush and are behind getting their jobs done. In order to achieve their tasks everyone is using a vehicle these days which has resulted in a society where around 25 million vehicles are being produced per year. This massive production and also the use of vehicles is not only degrading the environment by increasing the pollution but also this is resulting in increasing the risk and probability of fatality and destruction due to vehicle accidents. In these accidents most of the cases are mainly due to the reckless behaviour of the vehicle drivers which can have many reasons like restlessness for the driver etc. Road accidents have become very common nowadays. As more and more people are buying automobiles, the incidences of road accidents are just increasing day by day. Furthermore, people have also become more careless now. Not many people follow the traffic rules. Especially in big cities, there are various modes of transports. Moreover, the roads are becoming narrower and the cities have become more populated.

In this age of digitalisation everything that we do is trying to get done by machines and automation has become the main factor for development of the current generation. As a part of automation, self-driving cars emerged, became a keen interest, gained special focus and attention from big companies and already developed countries. But letting such a thing in the control of an automated software is never going to be an optimal or suggestable solution as a single failure can cause a lot of destruction. Hence rather than letting the machines control the things, they can assist the driver in having a safe and comfortable journey which can be a reliable solution. Most of the drivers who have a license are least bothered about traffic rules and do not abide by the message they get through traffic signs which they are ought to follow. Recklessness or lack of knowledge to understand the traffic sign can be the adding reasons to this behaviour. Hence a system that can recognise the traffic sign board and suggest the driver with appropriate messages can overcome this problem. The worst part of any accident that happens in the society is the pedestrians getting into trouble due to some other person's mistake. Hence a system that can detect a crowded area and assist the driver in knowing about the situation can help in saving lots of lives.

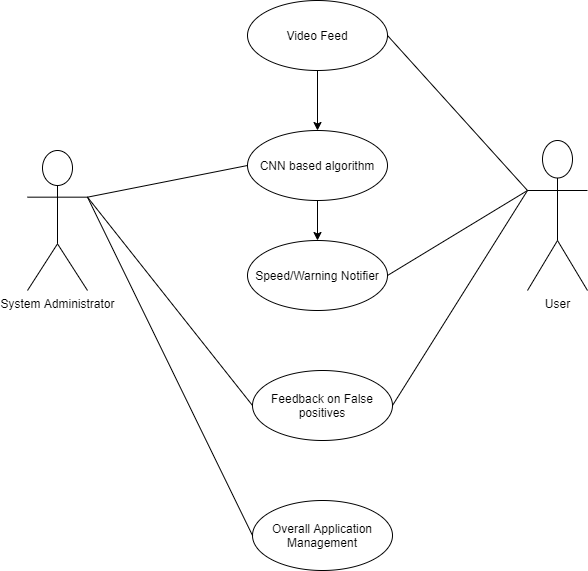
**Problem statement**

In order to ensure pedestrian safety and vehicle speed limits in areas with high pedestrian density/residential areas, a system to detect the traffic density and accordingly warn the driver about the speed limit for vehicles.

**Proposed Solution**

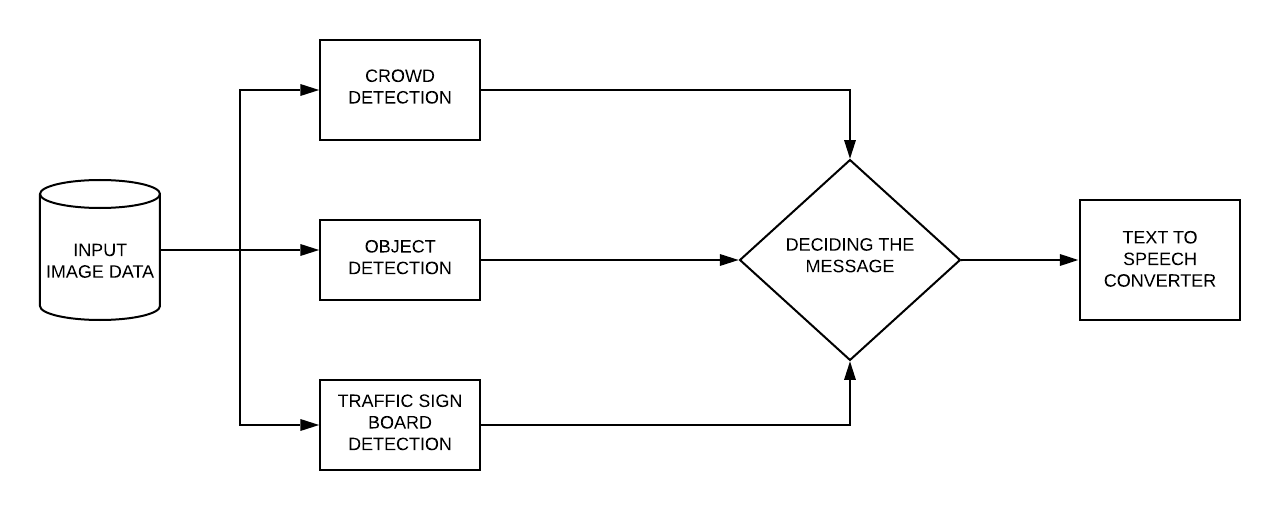
A software system that monitors the surroundings while driving, in real time by capturing the video of the surroundings through camera input and creates awareness about the situation to the driver by analysing the input it gets through the camera using the neural networking algorithms that have been developed and assists the user through relevant messages and appropriate signals that have been produce through decision made by the analysis.

**USE CASE DIAGRAM AND DESCRIPTION:**

****

* Video of the surroundings is captured through the camera that is available in the device.
* This video input is divided into frames per second.
* The frames produced are given as an input to the software where the image is processed by using image processing techniques
* These processed images are feeded to the trained models that have been saved after training a neural network. This produces an output giving the information about the image.
* This information is passed further to the decision process where the messages from all the processes are stored, which play a role in arriving at a relevant and suitable message to the driver.
* This message is passed to the process where the message is converted to speech and this audio is given as an output.

**Low level architecture:**

****

Implementation:

* Using the cv2 library the video input is captured and that input is converted to frames per second.
* These frames are further processed using image processing techniques and are passed into the processes which find the crowd intensity and recognize the traffic sign from the image.
* These processes produce results that are given as an input to the text to speech converter that produces audio messages which give assistance to the driver.
* CROWD INTENSITY DETECTION:

A pre-trained model from a research paper that uses a cascading algorithm to detect the persons is used. This pre trained model takes in the input image as an array and returns the coordinates of the region where the person is present and also the probability of how accurate the model is about that object being a person.

Using the coordinates a boundary box is drawn around the person if the probability that has been returned from the model is greater than the threshold probability and the count of the persons is incremented. If the count of the persons is greater than a number defined, then the message

‘Crowdy Area’ is produced and is given as an input to the text to speech converting engine to produce the output.

* TRAFFIC SIGN DETECTION:

The dataset required for training the model has been taken from the kaggle website. This data set is augmented and processed using image processing techniques. The input image is converted to grayscale image and finally the pixel range is normalised to a range of 0 - 1. These images are used for the training of the model. The convolution neural network has 64 2DCNN layers which have an activation function of relu for all the inner layers and each layer has an average pooling. All these layers are brought down to a dense layer of 43 nodes with an activation function softmax as there are 43 different classes as per the dataset available. Optimizer Adadelta is used while training the model. This model after training could achieve a validation accuracy of 96%. This model is saved into a pickle file which is further used in the process where it gets an input from the frame that has been taken as an input and the traffic sign is recognised and the accuracy of how probable that the image is a traffic sign is returned. Based on the accuracy being greater than the threshold value the message is produced which has the information about the traffic sign. This is given as an input to the text to speech converter and an output is generated accordingly.

Libraries used in the project:

import matplotlib.pyplot as plt

import numpy as np

import tensorflow as tf

import pandas as pd

import seaborn as sns

import pickle

import random

from sklearn.utils import shuffle

from tensorflow.keras import datasets,layers,models

from tensorflow import keras as hehe

Loading the Data using the Pickle Loader:

with open ("/content/drive/My Drive/trafficsigns/train.p",mode="rb") as training\_data:

  train = pickle.load(training\_data)

with open ("/content/drive/My Drive/trafficsigns/valid.p",mode="rb") as validation\_data:

  valid = pickle.load(validation\_data)

with open ("/content/drive/My Drive/trafficsigns/test.p",mode="rb") as testing\_data:

  test = pickle.load(testing\_data)

Our Project is categorized as:

* Converting the loaded images into gray scale images and then normalising them.
* Creating a two dimensional CNN.
* Compiling the CNN with the normalised data.

Converting the images into gray scale images:

x\_train\_gray = np.sum(x\_train/3 , axis=3 , keepdims = True)

x\_valid\_gray = np.sum(x\_valid/3 , axis = 3 , keepdims = True)

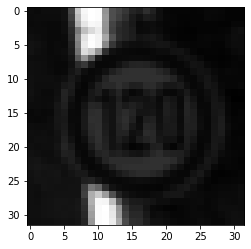
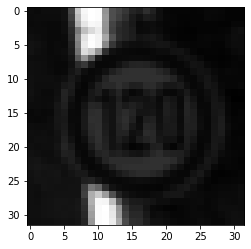
x\_test\_gray = np.sum(x\_test/3 , axis = 3 , keepdims =True)

Normalising the gray scale images:

x\_train\_gray\_norm  = (x\_train\_gray - 128)/128

x\_valid\_gray\_norm  = (x\_valid\_gray - 128)/128

x\_test\_gray\_norm  = (x\_test\_gray - 128)/128

a.Gray scale b.Original image c.Normalised

Creating a 2D CNN:

cnn = models.Sequential()

The **sequential** API allows one to create models layer-by-layer for most problems. It is limited in that it does not allow you to create models that share layers or have multiple inputs or outputs.As we have a single source for inputs we prefer sequential API over Functional API.

cnn.add(layers.Conv2D(16,(5,5),activation ='relu',input\_shape=(32,32,1)))

cnn.add(layers.AveragePooling2D())

The above code is used to create a layer with 16 filters .

The kernel size of (5 X 5) which is 2-tuple with the width and height of the 2D Convolution window is used.

RELU is taken as the activation function as it is considered to be as the

best for hidden layers of a CNN.

The input size contains the dimensional information of the input images.

Here it is specified as (32,32,1) as per the dimensions of the input image.

After applying a nonlinear activation function like RELU to the feature maps output by the CNN, adding a pooling layer to the CNN helps in down sample the feature maps by summarising the presence of features in patches by using strides or filters.

Average Pooling is used in this case to down sam-ple the features.

Similarly two other layers have been added :

cnn.add(layers.Conv2D(32,(5,5),activation ='relu',input\_shape=(32,32,1)))

cnn.add(MaxPooling2D(pool\_size=(2, 2)))

cnn.add(layers.Conv2D(64,(3,3),activation ='relu',input\_shape=(32,32,1)))

cnn.add(layers.AveragePooling2D())

cnn.add(layers.Flatten())

cnn.add(layers.Dense(120 , activation = 'relu'))

A layer with fully connected input and output nodes is considered to be a Dense layer. Here a dense layer with 120 filters and followed by an activation function RELU.

To avoid overfitting and not allow the model to memorize, Dropping out a few nodes is necessary and this can be achieved using the dropout function.

cnn.add(layers.Dropout(0.5))

All the hidden layers are followed by the RELU activation function whereas the output dense layer is followed by the SoftMax activation function.

Summary() Function is used to summarise the whole CNN that has been

constructed.

Model: "sequential\_9"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d\_10 (Conv2D) (None, 28, 28, 16) 416

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

average\_pooling2d\_9 (Average (None, 14, 14, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_11 (Conv2D) (None, 10, 10, 32) 12832

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 5, 5, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_12 (Conv2D) (None, 3, 3, 64) 18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

average\_pooling2d\_10 (Averag (None, 1, 1, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_8 (Flatten) (None, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_28 (Dense) (None, 120) 7800

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_5 (Dropout) (None, 120) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_29 (Dense) (None, 98) 11858

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_30 (Dense) (None, 84) 8316

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_31 (Dense) (None, 43) 3655

=================================================================

Total params: 63,373

Trainable params: 63,373

Non-trainable params: 0

To compile the constructed model compile function is used.

opt = hehe.optimizers.Adadelta(learning\_rate=0.01)

cnn.compile(optimizer = opt , loss = 'sparse\_categorical\_crossentropy' ,metrics = ['accuracy'])

history = cnn.fit(x\_train\_gray\_norm,y\_train,batch\_size = 50 ,epochs=40,verbose=1,validation\_data=(x\_valid\_gray\_norm,y\_valid))

Optimiser Adadelta is used and as the data is sparse with 43 output classes we use the loss function as sparse categorical cross entropy with accuracy as the metric for measurement.

To give the input normalised images into the CNN fit() function is used where batch size and the number of epochs are specified. The results produced by compiling the CNN are stored in a variable history.

Verbose is used to define how the user wants to see the output.

The last five epochs of the model are:

Epoch 36/40

696/696 [==============================] - 16s 23ms/step - loss: 0.2349 - accuracy: 0.9272 - val\_loss: 0.2879 - val\_accuracy: 0.9272

Epoch 37/40

696/696 [==============================] - 16s 24ms/step - loss: 0.2369 - accuracy: 0.9239 - val\_loss: 0.2877 - val\_accuracy: 0.9265

Epoch 38/40

696/696 [==============================] - 16s 23ms/step - loss: 0.2359 - accuracy: 0.9250 - val\_loss: 0.2883 - val\_accuracy: 0.9268

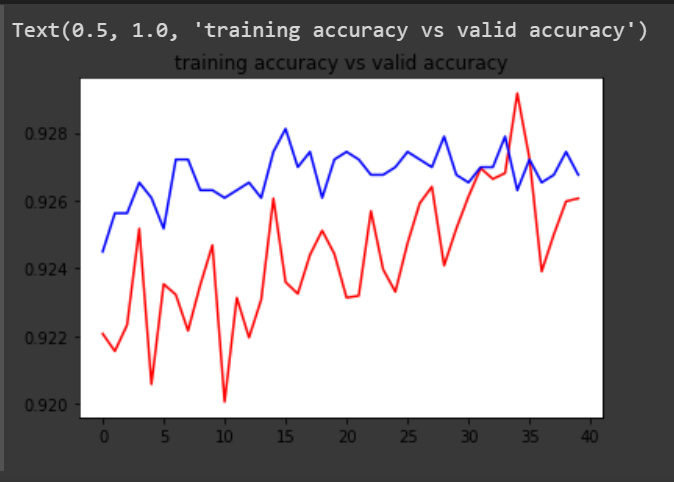
Epoch 39/40

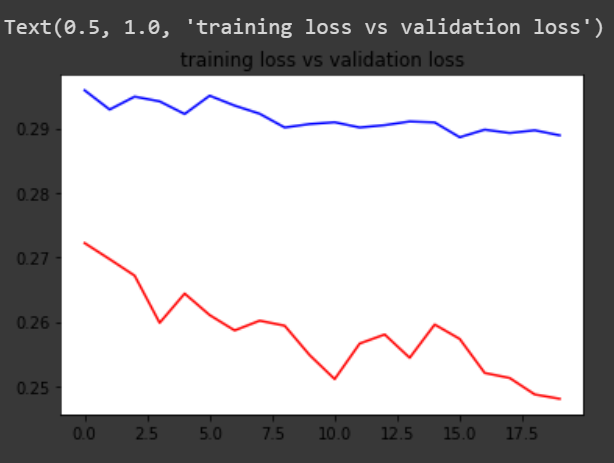
696/696 [==============================] - 16s 23ms/step - loss: 0.2366 - accuracy: 0.9260 - val\_loss: 0.2861 - val\_accuracy: 0.9274

Epoch 40/40

696/696 [==============================] - 16s 23ms/step - loss: 0.2331 - accuracy: 0.9261 - val\_loss: 0.2882 - val\_accuracy: 0.9268

History variable contains the values of accuracy and loss for training and validating the data during each epoch and these can be used to plot the graphs for visual assistance to evaluate the results produced.





To define the traffic sign based on the output that we get through the model a getclassname method is defined:

def getClassName(classNo):

if classNo == 0: return 'Speed Limit 20 km/h'

elif classNo == 1: return 'Speed Limit 30 km/h'

elif classNo == 2: return 'Speed Limit 50 km/h'

elif classNo == 3: return 'Speed Limit 60 km/h'

elif classNo == 4: return 'Speed Limit 70 km/h'

elif classNo == 5: return 'Speed Limit 80 km/h'

elif classNo == 6: return 'End of Speed Limit 80 km/h'

elif classNo == 7: return 'Speed Limit 100 km/h'

elif classNo == 8: return 'Speed Limit 120 km/h'

elif classNo == 9: return 'No passing'

elif classNo == 10: return 'No passing for vechiles over 3.5 metric tons'

elif classNo == 11: return 'Right-of-way at the next intersection'

elif classNo == 12: return 'Priority road'

elif classNo == 13: return 'Yield'

elif classNo == 14: return 'Stop'

elif classNo == 15: return 'No vechiles'

elif classNo == 16: return 'Vechiles over 3.5 metric tons prohibited'

elif classNo == 17: return 'No entry'

elif classNo == 18: return 'General caution'

elif classNo == 19: return 'Dangerous curve to the left'

elif classNo == 20: return 'Dangerous curve to the right'

elif classNo == 21: return 'Double curve'

elif classNo == 22: return 'Bumpy road'

elif classNo == 23: return 'Slippery road'

elif classNo == 24: return 'Road narrows on the right'

elif classNo == 25: return 'Road work'

elif classNo == 26: return 'Traffic signals'

elif classNo == 27: return 'Pedestrians'

elif classNo == 28: return 'Children crossing'

elif classNo == 29: return 'Bicycles crossing'

elif classNo == 30: return 'Beware of ice/snow'

elif classNo == 31: return 'Wild animals crossing'

elif classNo == 32: return 'End of all speed and passing limits'

elif classNo == 33: return 'Turn right ahead'

elif classNo == 34: return 'Turn left ahead'

elif classNo == 35: return 'Ahead only'

elif classNo == 36: return 'Go straight or right'

elif classNo == 37: return 'Go straight or left'

elif classNo == 38: return 'Keep right'

elif classNo == 39: return 'Keep left'

elif classNo == 40: return 'Roundabout mandatory'

elif classNo == 41: return 'End of no passing'

elif classNo == 42: return 'End of no passing by vechiles over 3.5 metric tons'

* INTEGRATING THE PROCESSES:

After getting the input image from the input camera, the processes that take this frame as input have to work simultaneously. To make them work simultaneously the procedure of multi threading is chosen over multiprocessing because multithreading uses a single memory unit for all the threads and also we have only a single image on which all these processes have to work. The first thread is used for processing the image ,detecting the crowd intensity and the traffic sign present in the image. The second thread is used to produce the audio output for the crowd intensity.

The third thread is used to produce the audio output for the traffic sign detected.

def detector():

global attt, stop\_thread

CLASSES = ["", "", "", "", "",

"", "", "", "", "", "", "",

"", "", "", "person", "", "",

"", "", ""]

color = np.random.uniform(200,200,200)

threshold = 0.999 # PROBABLITY THRESHOLD

font = cv2.FONT\_HERSHEY\_SIMPLEX

pickle\_in=open("S:\openlab\model\_trained.p","rb") ## rb = READ BYTE

model=pickle.load(pickle\_in)

def grayscale(img):

img = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

return img

def equalize(img):

img =cv2.equalizeHist(img)

return img

def preprocessing(img):

img = grayscale(img)

img = equalize(img)

img = img/255

return img

print("[INFO] loading model...")

net = cv2.dnn.readNetFromCaffe("Objects.prototxt.txt", "Objects.caffemodel")

print("[INFO] starting video stream...")

vs = VideoStream(src=0).start()

time.sleep(2.0)

fps = FPS().start()

personId = []

f = open("data.csv","w+",newline='')

# attt=0

t = time.strftime("%I:%M:%S")

t.strip("2019-12-11")

pltime=str(t)

pltime=pltime.split(':')

mm=int(pltime[1])

timecheck=mm

while True:

t = time.strftime("%I:%M:%S")

t.strip("2019-12-11")

pltime=str(t)

pltime=pltime.split(':')

hh=int(pltime[0])

mm=int(pltime[1])

pid = 0

personId.clear()

frame = vs.read()

frame = imutils.resize(frame, width=400)

(h, w) = frame.shape[:2]

blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),

0.007843, (300, 300), 127.5)

# predictions:

net.setInput(blob)

detections = net.forward()

img = np.asarray(frame)

img = cv2.resize(img, (32, 32))

img = preprocessing(img)

#cv2.imshow("Processed Image", img)

img = img.reshape(1, 32, 32, 1)

cv2.putText(frame, "CLASS: " , (20, 35), font, 0.45, (0, 0, 255), 1, cv2.LINE\_AA)

cv2.putText(frame, "PROBABILITY: ", (20, 75), font, 0.45, (0, 0, 255), 1, cv2.LINE\_AA)

# PREDICT IMAGE

predictions = model.predict(img)

classIndex = model.predict\_classes(img)

probabilityValue =np.amax(predictions)

if probabilityValue > threshold:

#print(getCalssName(classIndex))

cv2.putText(frame,str(classIndex)+" "+str(getCalssName(classIndex)), (120, 35), font, 0.75, (0, 0, 255), 2, cv2.LINE\_AA)

cv2.putText(frame, str(round(probabilityValue\*100,2) )+"%", (180, 75), font, 0.75, (0, 0, 255), 2, cv2.LINE\_AA)

sign = str(getCalssName(classIndex))

def speechhehe():

engine = pyttsx3.init()

engine.say(sign)

engine.runAndWait()

ssxthehe = threading.Thread(target=speechhehe)

ssxthehe.start()

for i in np.arange(0, detections.shape[2]):

confidence = detections[0, 0, i, 2]

if confidence > 0.2:

idx = int(detections[0, 0, i, 1])

box = detections[0, 0, i, 3:7] \* np.array([w, h, w, h])

(startX, startY, endX, endY) = box.astype("int")

#labeling the pesron with id

label = "id = {} {}: {:.2f}%".format(pid, CLASSES[idx],

confidence \* 100)

pid += 1

personId.append(pid)

personId.sort()

cv2.rectangle(frame, (startX, startY), (endX-100, endY),

color[idx], 2)

centerX = (startX+endX)//2

centerY = (startY+endY)//2

coord = (centerX, centerY)

y = startY - 15 if startY - 15 > 15 else startY + 15

cv2.putText(frame, label, (startX, y),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color[idx], 2)

#To FInd The Total

try:

attt+=personId[-1]

def speech():

engine = pyttsx3.init()

if(attt>4):

engine.say("Crowdy area")

engine.runAndWait()

ssxt\_ = threading.Thread(target=speech)

ssxt\_.start()

except IndexError:

pass

cv2.imshow("Frame", frame)

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

print("Final pid: "+str(personId))

break

# update the FPS counter

fps.update()

fps.stop()

print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))

print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

stop\_thread = True

cv2.destroyAllWindows()

vs.stop()

detect\_ = threading.Thread(target=detector)

detect\_.start()

* CONVERTING TEXT TO SPEECH:

To convert the text to speech pyttsx3 library is used. This has a predefined function that converts the text to speech offline without any use of the API’s that use the internet.

try:

attt+=personId[-1]

def speech():

engine = pyttsx3.init()

if(attt>4):

engine.say("Crowdy area")

engine.runAndWait()

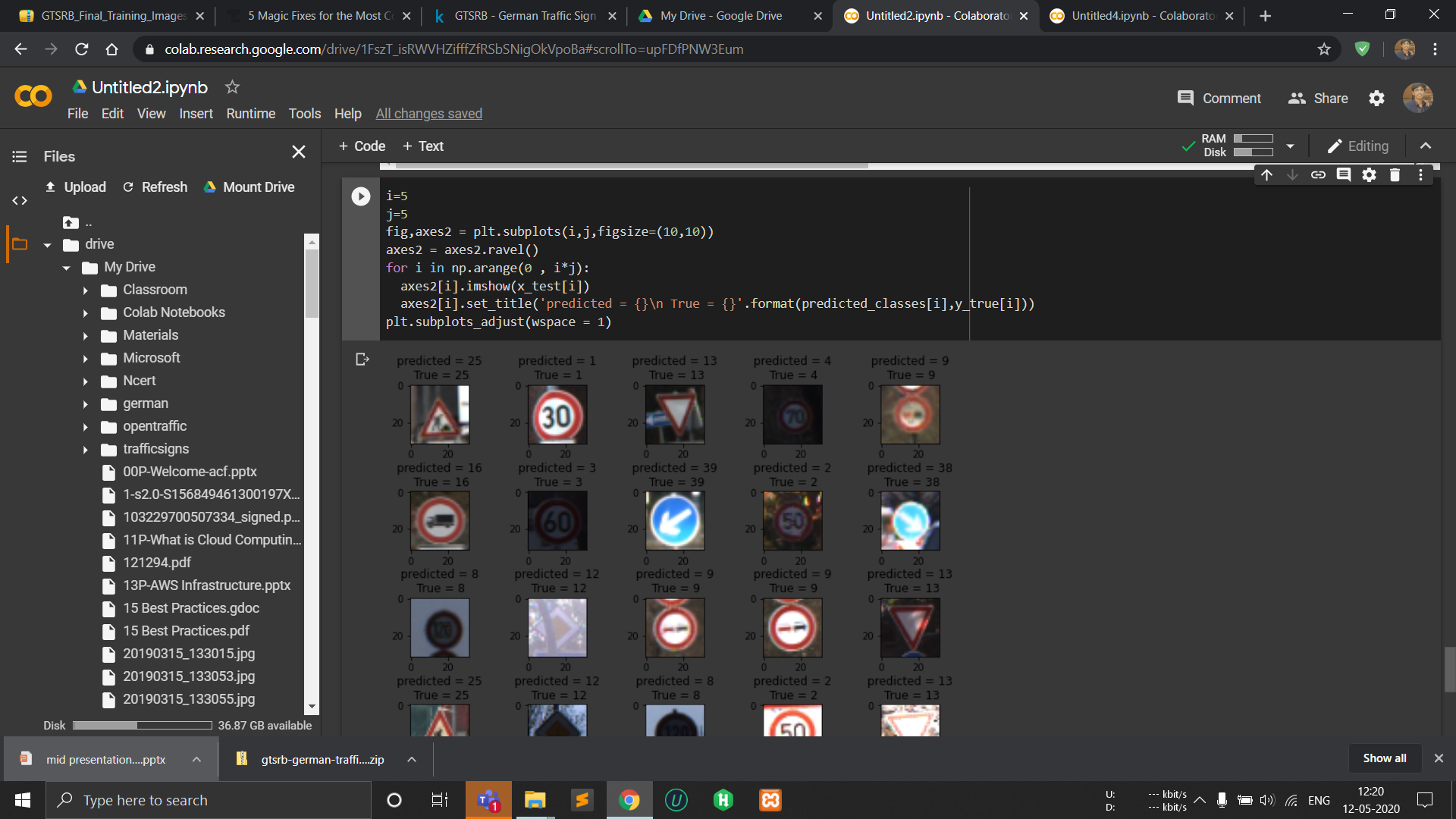
ssxt\_ = threading.Thread(target=speech)

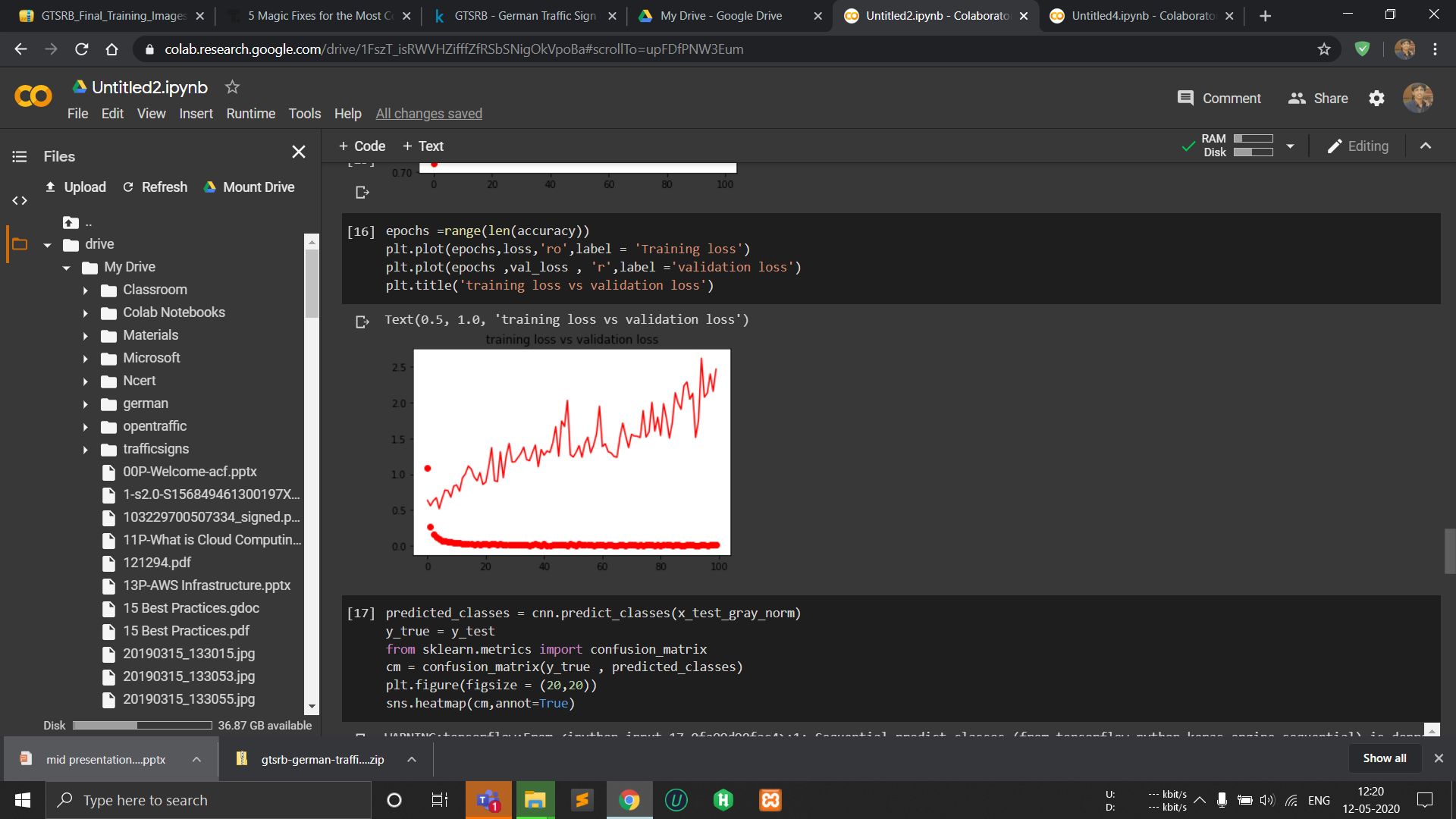
ssxt\_.start()

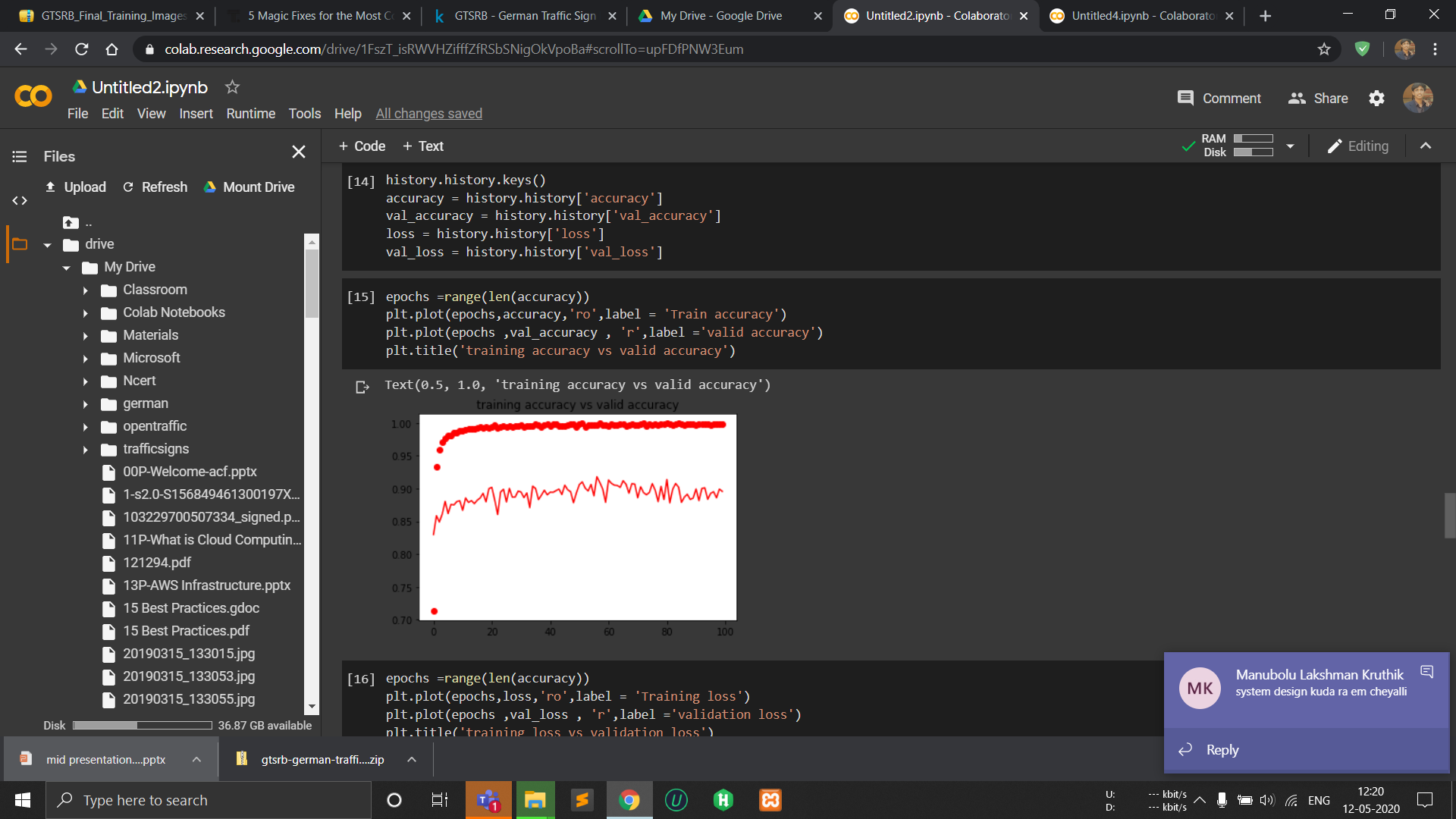
except IndexError:

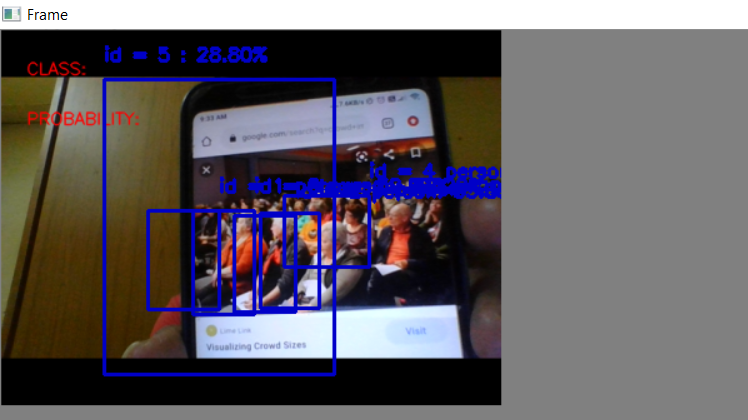
pass

RESULTS:

****







CONCLUSION

* The system can take the input from the device and produce appropriate messages.
* The System is able to do all these required processes in runtime.
* The messages are produced if there is a crowded area detected and if there is any traffic sign board recognised.

FURTHER SCOPE

* Developing the software furthermore in terms of accuracy and the range of classes that it can detect.
* Other functionalities like drowsy eye detection etc. can be added to make it more sophisticated.
* Producing a fully functioning embedded system that can be deployed onto the vehicles.
* An application that can be managed by the government to monitor the behavior of the drivers.
* Training the model used for traffic sign recognition with a functional API using input sources of images from various other standard traffic signs of countries in order to make it universal, so that this project can be integrated in any smart vehicle project.
* The range of capturing the image both in forward and sidewards can be increased and in order to increase the accuracy better image processing techniques like canny edge detection etc. can be used.

REFERENCES

* Concept for traffic sign board recognition - <https://www.hindawi.com/journals/mpe/2015/250461/>
* keras documentation for producing the networks- <https://keras.io/guides/sequential_model/>
* Using CNN iver RCNN and other networks

<https://arxiv.org/pdf/1311.2524.pdf>

<https://arxiv.org/pdf/1504.08083.pdf>

* Using pyttsx3-

<https://pypi.org/project/pyttsx3/>

* Using multithreading concepts over multiprocessing -

<https://docs.python.org/3/library/threading.html>

* Pre trained model -

<https://paperswithcode.com/task/crowd-counting>