TRAFFIC SIGNS CLASSIFIER USING NEURAL NETWORKS

A

Mini Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

By

AKASH.S.VORA

1602-19-733-126

BAKSHI ABHINITH

1602-19-733-124

Department of Computer Science & Engineering

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2021

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

DECLARATION BY THE CANDIDATE

I, **AKASH S VORA,** bearing hall ticket number, **1602-19-733-126**, hereby declare that the project report entitled **"TRAFFIC SIGNS CLASSIFIER USING NEURAL NETWORKS"** Department of Computer Science & Engineering, VCE, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Computer Science & Engineering**.

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

AKASH S VORA,

1602-19-733-126.

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

BONAFIDE CERTIFICATE

This is to certify that the project entitled **“TRAFFIC SIGNS CLASSIFIER USING NEURAL NETWORKS”** being submitted by **AKASH S VORA,** bearing **1602-19-733-126,** in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering is a record of bonafide work carried out by him/her under my guidance.

Ms. T. Nishitha

Assistant Professor

Dept. of CSE

ACKNOWLEDGEMENT

With immense pleasure, we record our deep sense of gratitude to our guide Ms. T. Nishitha, Assistant Professor, Vasavi College of Engineering, Hyderabad, for the valuable guidance and suggestions, keen interest and thorough encouragement extended throughout the period of the project work. I consider myself lucky enough to be part of this project. This project would add as an asset to my academic profile.

We express our thanks to all those who contributed for the successful completion of our project work.

ABSTRACT

In the world of Artificial Intelligence and advancement in technologies, many researchers and big companies like Tesla, Uber, Google, Mercedes-Benz, Toyota, Ford, Audi, etc are working on autonomous vehicles and self-driving cars. One of the many important aspects of a self-driving car is its ability to detect traffic signs in order to provide safety and security for the people not only inside the car but also outside it. So, for achieving accuracy in this technology, the vehicles should be able to interpret traffic signs and make decisions accordingly.

We have focused our project on the German traffic signs. We used the GTSRB traffic sign dataset. The dataset contains about 40,000 images of different traffic signs. It is further classified into 43 different classes. The dataset is quite varying, some of the classes have many images while some classes have few images.

TABLE OF CONTENTS

Page No.

LIST OF FIGURES

INTRODUCTION

Traffic-sign detection and classification is an interesting topic in computer-vision and it is especially important in the field of autonomous vehicle technology. Realtime traffic-sign detection algorithms have to be deployed in self-driving cars in order to become a commonplace in the roads of the future.

The main objective of our project is to design a computer based system which can automatically detect the road signs so as to provide assistance to the user or the machine so that they can take appropriate actions. Our approach consists of building a model using convolutional neural networks by extracting traffic signs from an image using Various filters. We have used convolutional neural networks (CNN) to classify the traffic signs.

2.1 OVERVIEW

There are several different types of traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc., Traffic signs classification is the process of identifying which class a traffic sign belongs to.

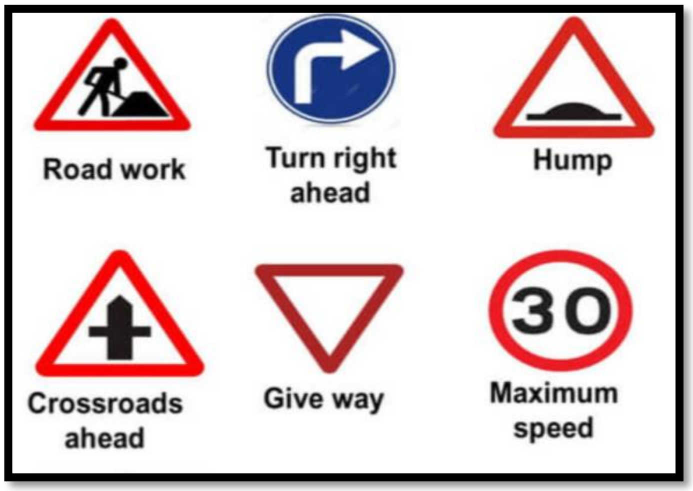


Figure 2.1.1

Here, we have built a deep neural network model that can classify traffic signs present in an image into different categories (43 categories). With this model, we were able to read and understand traffic signs which are a very important task for all autonomous vehicles.

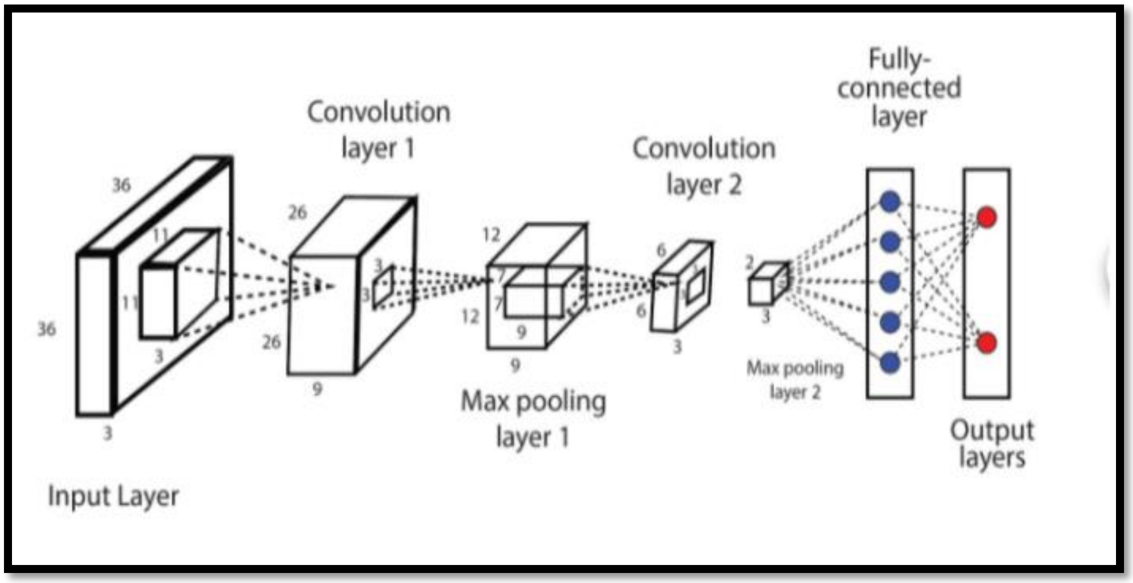


Figure 2.1.2

Firstly we have explored the dataset and found that there are a total of 43 classes of traffic signs. The next step in our approach is building a Convolutional Neural Network (CNN) model that will then be used for classifying the given traffic signs. A Convolutional Neural Network (CNN) is a type of neural network model which allows us to extract higher representations for images. Unlike the classical image recognition where we define the image features ourselves, CNN takes the image's raw pixel data, trains the model, then extracts the features automatically for better classification.

Our model consists of two main layers, The convolution layer and the pooling layer. Every step in a convolution layer sweeps the window through images then calculates its input and filter dot product pixel values. The use of pooling layer is to reduce the spatial dimension of the input volume for next layers. Note that it only affects weight and height but not depth. The max pool layer is similar to convolution layer, but instead of doing a convolution operation, we select the max values in the receptive fields of the input, saving the indices and then producing a summarized output volume.

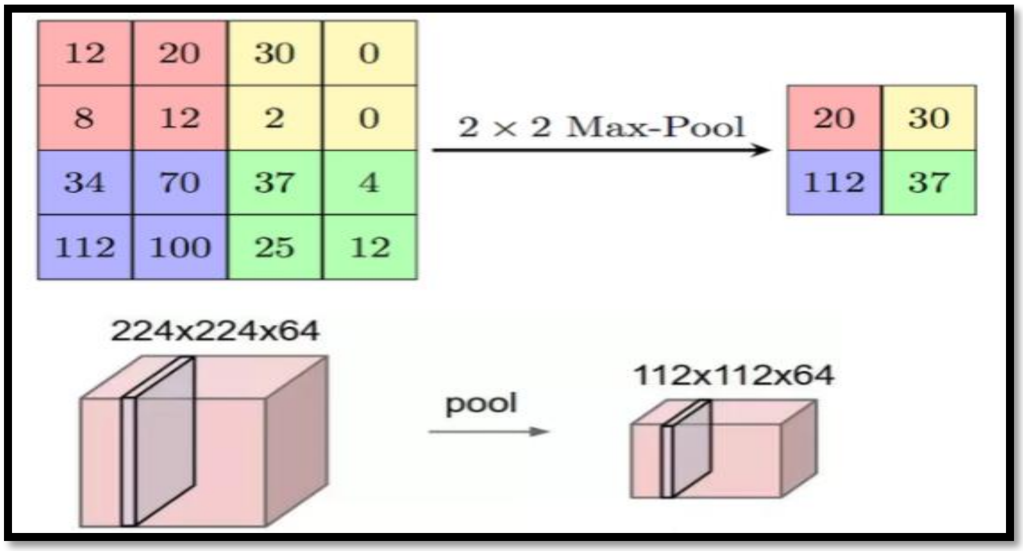


Figure 2.1.3

Once the desired model is built, we then train the data with our training dataset. After the training was done for a various set of values, we tuned

Our model to achieve the maximum accuracy of 95.6% on the training data.

Then we tested our model with test dataset and the accuracy we obtained was 95%.

We have also built a GUI in which we have used the saved model (.h5 file) to classify the images that were uploaded into the interface.

2.2 ABOUT THE DATASET

The German Traffic Sign Recognition Benchmark (GTSRB) contains 43 classes of traffic signs, split into 39,209 training images and 12,630 test images. The images have varying light conditions and rich backgrounds. The dataset is quite varying, some of the classes have many images while some classes have few images.

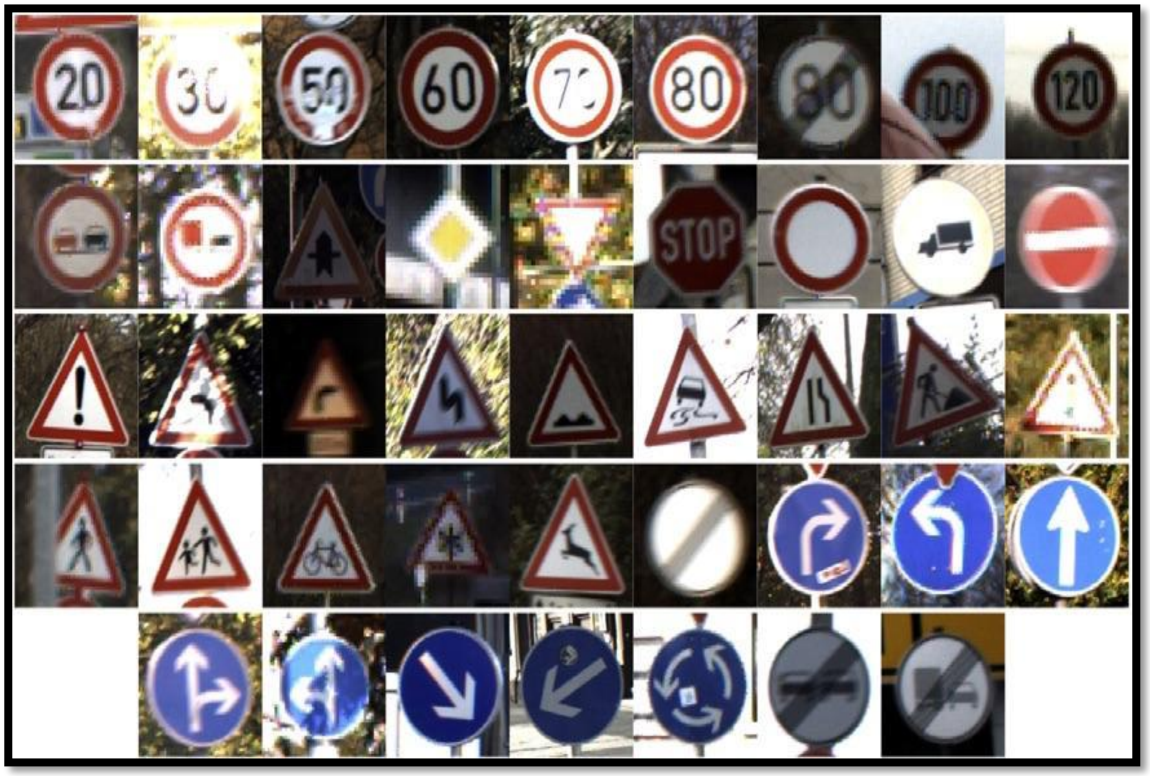


Figure 2.2.1

2.3 OBJECTIVE

The main objective of our project is to design a model which can automatically classify the road signs so as to assist the user or the machine in taking appropriate actions. Our approach consists of building a model using convolutional neural networks by extracting traffic signs from an image using various filters. We have used convolutional neural networks (CNN) to classify the traffic signs.

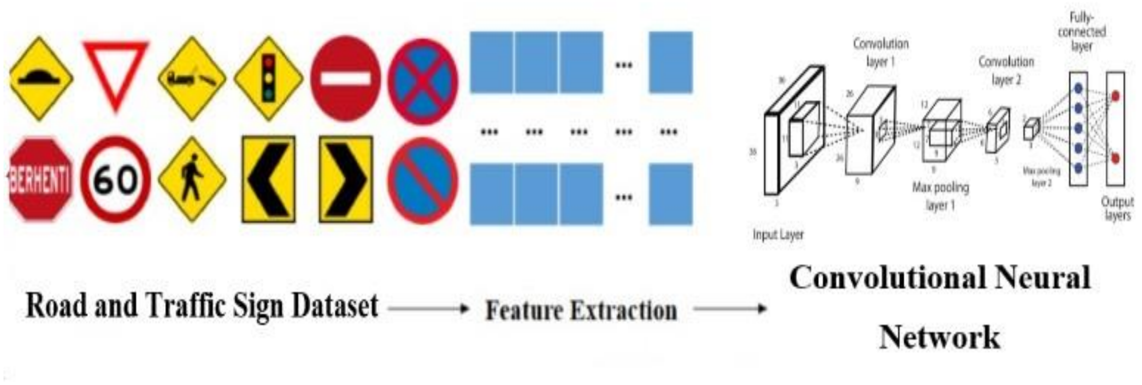


Figure 2.3.1

SYSTEM REQUIREMENTS

Hardware:

- Minimum RAM required: 4 GB
- Input devices: Mouse, Keyboard
- Output devices: Monitor

Software:

- Google Colab
- Python 3.8.3
- Windows 8.1 or above

IMPLEMENTATION

• Model training and testing

```
!pip install tensorflow

!pip install np_utils


import keras.utils

import np_utils


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

import keras

from sklearn.model_selection import train_test_split


from keras.models import Sequential, load_model

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

```
""" IMPORTANT """


from tensorflow.keras.utils import to_categorical


from zipfile import ZipFile
file_name = '/content/drive/MyDrive/Untitled folder/archive.zip'
with ZipFile(file_name,'r') as zip:
  zip.extractall()
  print("Extraction completed")


from google.colab import drive
drive.mount('/content/drive')


data = []
Class = []
no_of_classes = 43
cur_path = '/content/'


for i in range(no_of_classes):
    path = os.path.join(cur_path,'train',str(i))
    all_images = os.listdir(path)
    for a in all_images:
          try:
            imageRGB = Image.open(path + '/'+ a)

            imageRGB = imageRGB.resize((30,30))

            imageRGB = np.array(imageRGB)

            data.append(imageRGB)

            Class.append(i)

        except:

            print("Error")


""" ARRAY OF IMAGES """


data = np.array(data)

Class = np.array(Class)


X_train, X_test, y_train, y_test = train_test_split(data, Class, test_size=0.2,
random_state=42)


y_train = to_categorical(y_train, 43)

y_test = to_categorical(y_test, 43)

print("successful")


""" MODEL CREATION """


model = Sequential()
```

```
#First

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))


#Second

#model.add(Conv2D(filters=64, kernel_size=(5,5), activation='relu',
input_shape=X_train.shape[1:]))


#model.add(Conv2D(filters=64, kernel_size=(5,5), activation='relu'))


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


#model.add(Dropout(rate=0.20))


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=0.20))

model.add(Flatten())

model.add(Dense(512, activation='relu'))

model.add(Dropout(rate=0.6))

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

print("Model created successfully")


""" TRAINING THE MODEL """


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

modelFitting = model.fit(X_train, y_train, batch_size=64, epochs = 20 ,
validation_data=(X_test, y_test))

print("Model trained successfully")


model.save("trained_model_new.h5")

model.summary()


""" TEST ACCURACY """


from sklearn.metrics import accuracy_score
```

```
y_test = pd.read_csv('/content/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)


#Accuracy with the test data


from sklearn.metrics import accuracy_score

import warnings

warnings.filterwarnings('ignore')

print("Accuracy : "+str(accuracy_score(labels, pred)*100))


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

labels = y_test["ClassId"].values

print(accuracy_score(labels, pred))

print(type(pred))

print(pred)
```

```
""" CUSTOM TEST """

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' }
print("Classes initialized")
```

```
model = tf.keras.models.load_model('/content/trained_model_new.h5')

model.summary()


image = Image.open('/content/00035.png')

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

image = np.expand_dims(image, axis=0)

image = np.array(image)

pred = model.predict_classes([image])[0]

sign = classes[pred+1]

print(sign)

im = Image.open('/content/00035.png')

im


image = Image.open('/content/stop.png')

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

image = np.expand_dims(image, axis=0)

image = np.array(image)

pred = model.predict_classes([image])[0]

sign = classes[pred+1]

print(sign)

im = Image.open('/content/stop.png')

im
```

```
""" GRAPH FOR THE LOSS """


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()


""" GRAPH FOR ACCURACY """

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()
```

- **GUI Implementation**

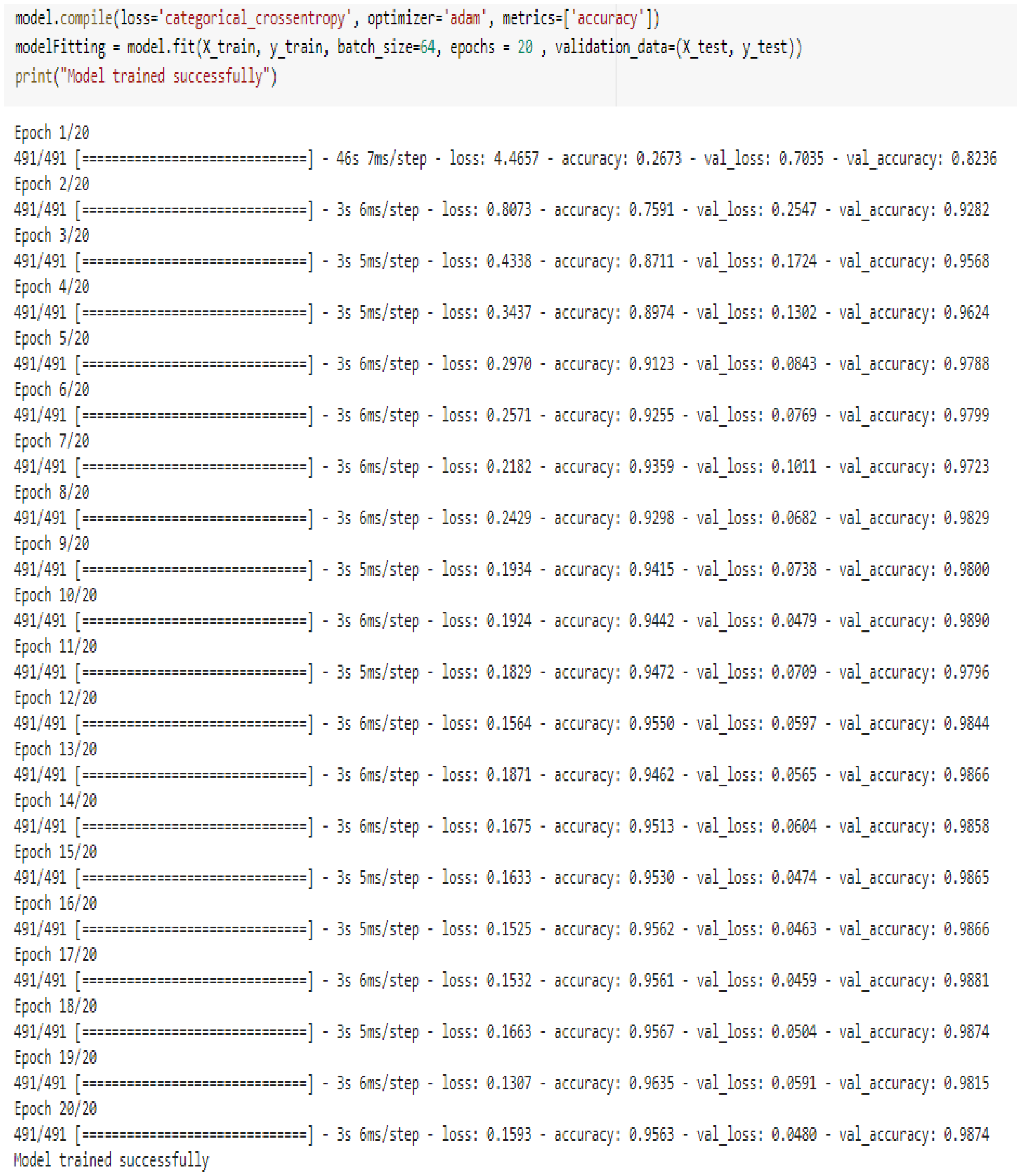
```
#tkinter - used as a standard GUI library in Python

import tkinter as tk

#tkinter.filedialog - offers a set of dialogs when working with files

from tkinter import filedialog

from tkinter import *

#Python Image Library (PIL) - ImageTk contains support to create and modify
PhotoImage objects from PIL images

from PIL import ImageTk, Image

import numpy

#used to load the trained model to classify various traffic signs

from keras.models import load_model

model = load_model('trained_model_new.h5')


#The classes dictionary is used to label all the classes of various traffic signs.

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' }


#Initialise the GUI

top = tk.Tk()

top.geometry('960x500')

top.title('Traffic Sign Classification Project')

top.configure(background='#99CCFF')


#Setting the background image

bg = ImageTk.PhotoImage(Image.open("background.jpg"))

label1 = Label( top, image = bg)
```

```
label1.place(x = 0, y = 0)


label = Label(top,background='#99CCFF', font=('arial',15,'bold'))

sign_image = Label(top)


def classify(file_path):

    global label_packed

    #Opens the image in the corresponding file path

    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="Classify Image",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:
        print("Please upload an image!!")
        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="Traffic Signs Classifier",pady=20, font=('arial',20,'bold'))
heading.configure(background='#FFFFCC',foreground='#364156')
heading.pack()      #side = TOP is set as default

#Execute Tkinter
top.mainloop()
```

OUTPUT OF THE PROGRAM

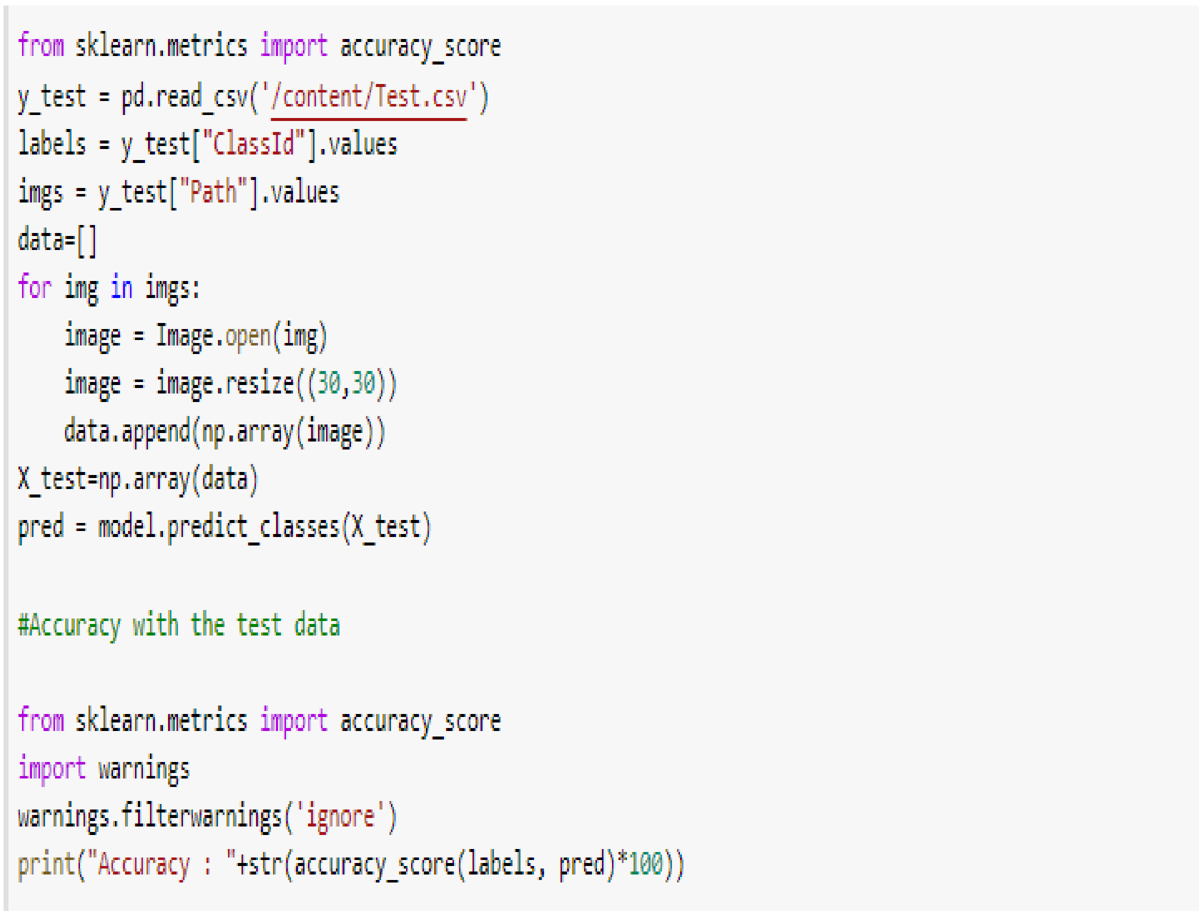


```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
modelFitting = model.fit(X_train, y_train, batch_size=64, epochs = 20 , validation_data=(X_test, y_test))
print("Model trained successfully")
```

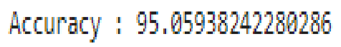
```
Epoch 1/20
491/491 [==============================] - 46s 7ms/step - loss: 4.4657 - accuracy: 0.2673 - val_loss: 0.7035 - val_accuracy: 0.8236
Epoch 2/20
491/491 [==============================] - 3s 6ms/step - loss: 0.8073 - accuracy: 0.7591 - val_loss: 0.2547 - val_accuracy: 0.9282
Epoch 3/20
491/491 [==============================] - 3s 5ms/step - loss: 0.4338 - accuracy: 0.8711 - val_loss: 0.1724 - val_accuracy: 0.9568
Epoch 4/20
491/491 [==============================] - 3s 5ms/step - loss: 0.3437 - accuracy: 0.8974 - val_loss: 0.1302 - val_accuracy: 0.9624
Epoch 5/20
491/491 [==============================] - 3s 6ms/step - loss: 0.2970 - accuracy: 0.9123 - val_loss: 0.0843 - val_accuracy: 0.9788
Epoch 6/20
491/491 [==============================] - 3s 6ms/step - loss: 0.2571 - accuracy: 0.9255 - val_loss: 0.0769 - val_accuracy: 0.9799
Epoch 7/20
491/491 [==============================] - 3s 6ms/step - loss: 0.2182 - accuracy: 0.9359 - val_loss: 0.1011 - val_accuracy: 0.9723
Epoch 8/20
491/491 [==============================] - 3s 6ms/step - loss: 0.2429 - accuracy: 0.9298 - val_loss: 0.0682 - val_accuracy: 0.9829
Epoch 9/20
491/491 [==============================] - 3s 5ms/step - loss: 0.1934 - accuracy: 0.9415 - val_loss: 0.0738 - val_accuracy: 0.9800
Epoch 10/20
491/491 [==============================] - 3s 6ms/step - loss: 0.1924 - accuracy: 0.9442 - val_loss: 0.0479 - val_accuracy: 0.9890
Epoch 11/20
491/491 [==============================] - 3s 5ms/step - loss: 0.1829 - accuracy: 0.9472 - val_loss: 0.0709 - val_accuracy: 0.9796
Epoch 12/20
491/491 [==============================] - 3s 6ms/step - loss: 0.1564 - accuracy: 0.9550 - val_loss: 0.0597 - val_accuracy: 0.9844
Epoch 13/20
491/491 [==============================] - 3s 6ms/step - loss: 0.1871 - accuracy: 0.9462 - val_loss: 0.0565 - val_accuracy: 0.9866
Epoch 14/20
491/491 [==============================] - 3s 6ms/step - loss: 0.1675 - accuracy: 0.9513 - val_loss: 0.0604 - val_accuracy: 0.9858
Epoch 15/20
491/491 [==============================] - 3s 5ms/step - loss: 0.1633 - accuracy: 0.9530 - val_loss: 0.0474 - val_accuracy: 0.9865
Epoch 16/20
491/491 [==============================] - 3s 5ms/step - loss: 0.1525 - accuracy: 0.9562 - val_loss: 0.0463 - val_accuracy: 0.9866
Epoch 17/20
491/491 [==============================] - 3s 6ms/step - loss: 0.1532 - accuracy: 0.9561 - val_loss: 0.0459 - val_accuracy: 0.9881
Epoch 18/20
491/491 [==============================] - 3s 5ms/step - loss: 0.1663 - accuracy: 0.9567 - val_loss: 0.0504 - val_accuracy: 0.9874
Epoch 19/20
491/491 [==============================] - 3s 6ms/step - loss: 0.1307 - accuracy: 0.9635 - val_loss: 0.0591 - val_accuracy: 0.9815
Epoch 20/20
491/491 [==============================] - 3s 6ms/step - loss: 0.1593 - accuracy: 0.9563 - val_loss: 0.0480 - val_accuracy: 0.9874
Model trained successfully
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_4 (Conv2D)            (None, 26, 26, 32)        2432      
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 22, 22, 32)        25632     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 11, 11, 32)        0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 11, 11, 32)        0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 9, 9, 64)          18496     
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 7, 7, 64)          36928     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 64)          0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 3, 3, 64)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 512)               295424    
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 43)                22059     
=================================================================
Total params: 400,971
Trainable params: 400,971
Non-trainable params: 0
_________________________________________________________________
```



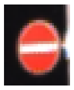
```
from sklearn.metrics import accuracy_score
y_test = pd.read_csv('/content/Test.csv')
labels = y_test["ClassId"].values
imgs = y_test["Path"].values
data=[]
for img in imgs:
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))
X_test=np.array(data)
pred = model.predict_classes(X_test)

#Accuracy with the test data

from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
print("Accuracy : "+str(accuracy_score(labels, pred)*100))
```



```
Accuracy : 95.05938242280286
```

```
image = Image.open('/content/00035.png')
image = image.resize((30,30))
image = np.expand_dims(image, axis=0)
image = np.array(image)
pred = model.predict_classes([image])[0]
sign = classes[pred+1]
print(sign)
im = Image.open('/content/00035.png')
im
```

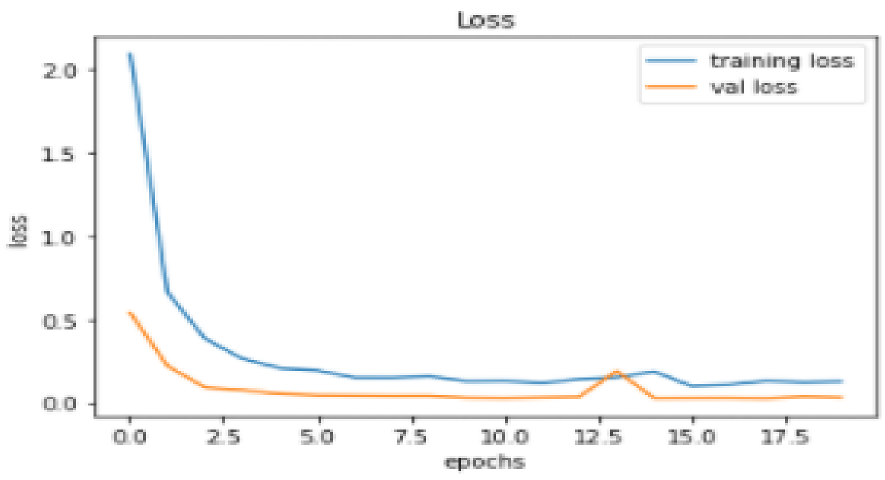
```
No entry
```



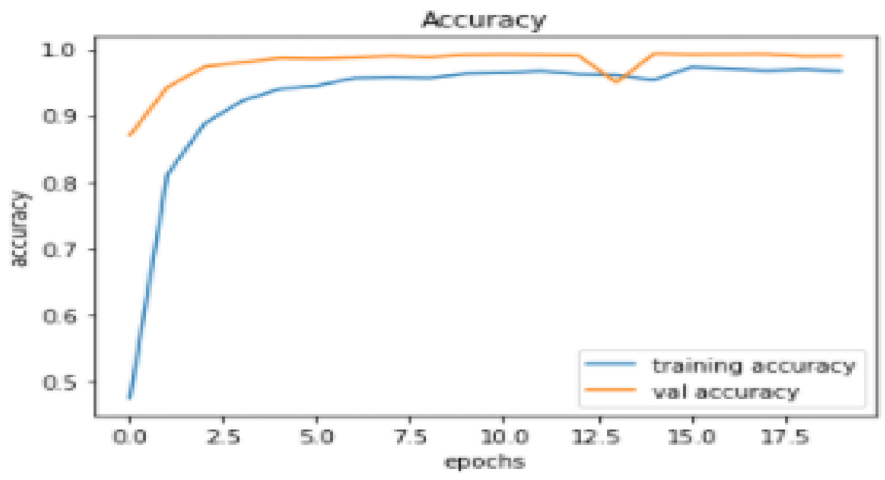
```
image = Image.open('/content/stop.png')
image = image.resize((30,30))
image = np.expand_dims(image, axis=0)
image = np.array(image)
pred = model.predict_classes([image])[0]
sign = classes[pred+1]
print(sign)
im = Image.open('/content/stop.png')
im
```

```
Speed limit (50km/h)
```

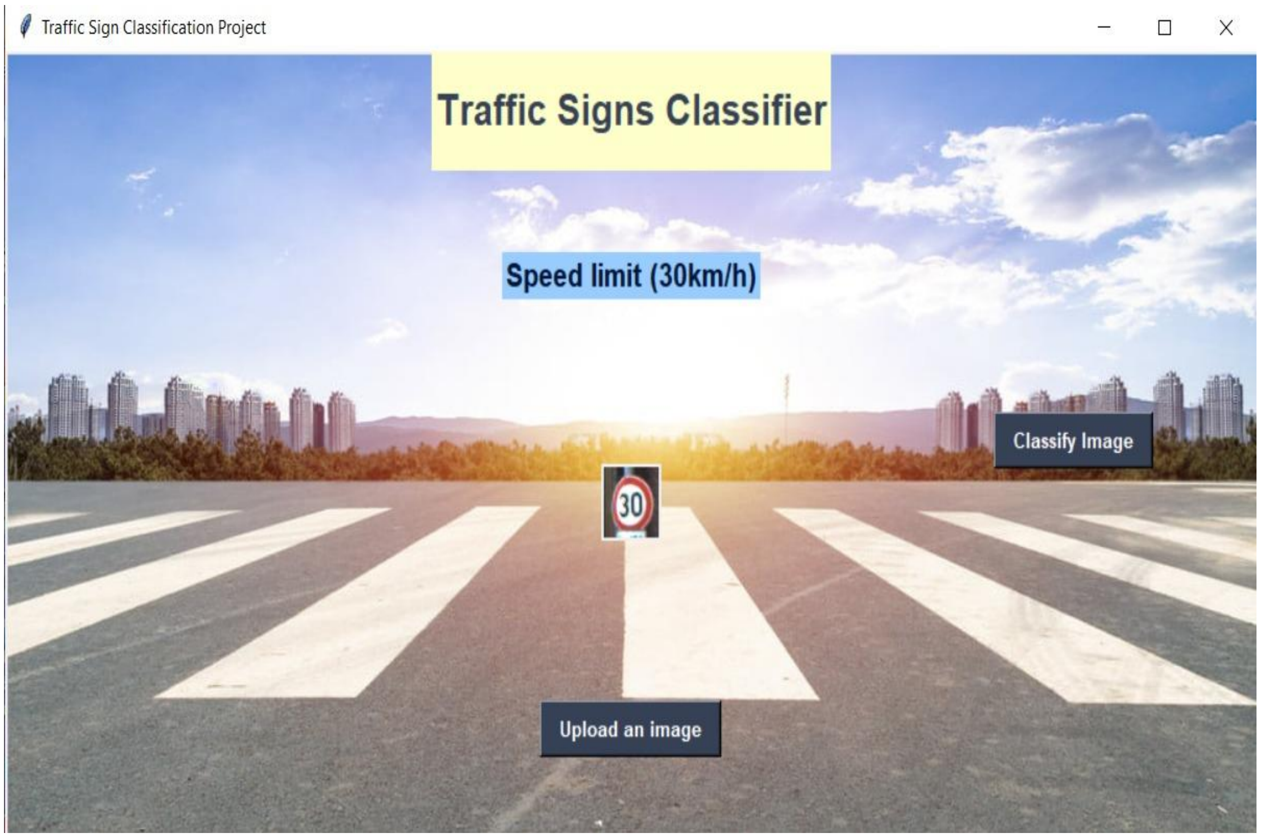
```
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



```
# accuracy
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()
```



❖ GUI


Traffic Sign Classification Project
Traffic Signs Classifier
Speed limit (30km/h)
Classify Image
30
Upload an image

CONCLUSION AND FUTURE WORK

The main aim of this project was to classify traffic signs using Convolutional neural networks, which has been achieved.

We would like to extend this project by implementing this model in real time by building a small prototype that captures high resolution images at average speeds which will be classified which can then be used to assist a driver, in the case of a manual car, or a machine, in the case of an autonomous vehicle.

REFERENCES

- Uday Kamal, Sowmitra Das, Abid Abrar and Md. Kamrul Hasan, “Traffic-Sign Detection and Classification Under Challenging Conditions: A Deep Neural Network Based Approach”, September 2017.
- https://data-flair.training/
- https://www.geeksforgeeks.org/
- https://www.analyticsvidhya.com/blog/
- https://www.analyticssteps.com/
- https://stackoverflow.com/
- https://www.youtube.com/